CoNLL-2009

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL)

Conference Chairs:
Suzanne Stevenson and Xavier Carreras

June 4–5, 2009
Boulder, Colorado

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

ISBN 978-1-932432-29-9

Introduction

The 2009 Conference on Computational Natural Language Learning is the thirteenth in the series of annual meetings organized by SIGNLL, the ACL special interest group on natural language learning. CoNLL-2009 will be held in Boulder, CO, 4–5 June 2009, in conjunction with NAACL HLT.

For our special focus this year in the main session of CoNLL, we invited papers on unsupervised, minimally supervised and semi-supervised methods in natural language learning, as well as on incremental learning methods. As with earlier CoNLLs, we encouraged papers that addressed these issues from the perspective both of human language acquisition and of NLP systems.

We received 70 submissions to the main session on these and other relevant topics, of which 11 were withdrawn. Of the remaining 59 papers, 15 were selected to appear in the conference program as oral presentations, and 10 were chosen as posters. All accepted papers appear here in the proceedings.

Our invited speakers reflect the state-of-the-art in human and machine learning of natural language, and we are grateful to Michael Frank and Andrew McCallum for agreeing to speak on their exciting new work in these areas.

As in previous years, CoNLL-2009 has a shared task, Syntactic and Semantic Dependencies in Multiple Languages. This is an extension of the CoNLL-2008 shared task to multiple languages (English plus Catalan, Chinese, Czech, German, Japanese and Spanish). Among the new features are compatible evaluation for several languages and their comparison, and learning curves for languages with large datasets. We expect that this major comparative exercise will lead to very enlightening results and discussion that will serve to move the field forward. The Shared Task papers are collected into an accompanying volume of CoNLL-2009. We thank Jan Hajic and the rest of the organizers for their great effort in running the Shared Task.

We would like to thank the members of the SIGNLL steering committee for useful discussion, especially Lluís Màrquez and Joakim Nivre, who helped us greatly with advice around the conference organization, and Erik Tjong Kim Sang, who acted as the information officer. We also appreciate the help we received from NAACL HLT organizers, including Martha Palmer, Mark Hasegawa-Johnson, Nizar Habash, Christy Doran, Eric Ringger, and Priscilla Rasmussen.

Finally, many thanks to Google for sponsoring the best paper award at CoNLL-2009.

We hope you find CoNLL-2009 a fruitful venue for discussion and interaction on the exciting topics covered by our program.

Suzanne Stevenson and Xavier Carreras

2009 Conference Chairs

Conference Chairs:

Suzanne Stevenson, University of Toronto
Xavier Carreras, MIT

Program Committee:

Steve Abney, University of Michigan
Afra Alishahi, Saarland University
Galen Andrew, Microsoft Research
Giuseppe Attardi, University of Pisa
Kirk Baker, Collexis Inc
Timothy Baldwin, University of Melbourne
Roberto Basili, University of Roma, Tor Vergata
Phil Blunsom, University of Edinburgh
S.R.K. Branavan, MIT
Chris Brew, Ohio State University
Sabine Buchholz, Toshiba Research Europe Ltd.
Paula Buttery, Cambridge University
Nicola Cancedda, Xerox Research Center Europe
Sander Canisius, The Netherlands Cancer Institute
Claire Cardie, Cornell University
Ming-Wei Chang, University of Illinois at Urbana-Champaign
Ciprian Chelba, Google
Colin Cherry, Microsoft Research
Massimiliano Ciaramita, Google Research Zurich
Alexander Clark, Royal Holloway, University of London
Stephen Clark, University of Cambridge
James Clarke, University of Illinois at Urbana-Champaign
Michael Collins, MIT
Paul Cook, University of Toronto
James Cussens, University of York
Walter Daelemans, University of Antwerp
Hal Daume III, University of Utah
Jacob Eisenstein, University of Illinois at Urbana-Champaign
Katrin Erk, University of Texas at Austin
Anna Feldman, Montclair State University
Jenny Finkel, Stanford University
Radu Florian, IBM Research
Dayne Freitag, SRI International
Pascale Fung, Hong Kong University of Science and Technology
Michel Galley, Stanford University
Daniel Gildea, University of Rochester

Roxana Girju, University of Illinois at Urbana-Champaign
John Hale, Cornell University
James Henderson, University of Geneva
Julia Hockenmaier, University of Illinois at Urbana-Champaign
Liang Huang, Google Research
Richard Johansson, University of Trento
Rie Johnson (formerly, Ando), RJ Research Consulting
Rohit Kate, University of Texas at Austin
Philipp Koehn, University of Edinburgh
Rob Koeling, Sussex University
Anna Korhonen, Cambridge University
Shalom Lappin, King's College, London
Dekang Lin, Google Research
Xiaofei Lu, Pennsylvania State University
Rob Malouf, San Diego State University
Yuji Matsumoto, Nara Institute of Science and Technology
Diana McCarthy, University of Sussex
Ryan McDonald, Google Research
Paola Merlo, University of Geneva
Rada Mihalcea, University of North Texas
Saif Mohammad, University of Maryland
Alessandro Moschitti, University of Trento
Gabriele Musillo, University of Geneva
John Nerbonne, University of Groningen
Hwee Tou Ng, National University of Singapore
Vincent Ng, University of Texas at Dallas
Grace Ngai, The Hong Kong Polytechnic University
Joakim Nivre, Uppsala University and Växjö University
Franz Och, Google Research
Miles Osborne, University of Edinburgh
Chris Parisien, University of Toronto
Ted Pedersen, University of Minnesota, Duluth
Slav Petrov, UC Berkeley
David Powers, Flinders University
Chris Quirk, Microsoft Research
Ari Rappoport, Hebrew University
Lev Ratinov, University of Illinois at Urbana-Champaign
Sebastian Riedel, University of Tokyo
Brian Roark, Oregon Health & Science University
Dan Roth, University of Illinois at Urbana-Champaign
William Sakas, City University of New York
Sabine Schulte im Walde, University of Stuttgart
Libin Shen, BBN Technologies
David Smith, University of Massachusetts Amherst
Noah Smith, Carnegie Mellon University
Benjamin Snyder, MIT

Caroline Sporleder, Saarland University
Richard Sproat, Oregon Health & Science University
Mihai Surdeanu, Stanford University
Ivan Titov, University of Illinois at Urbana-Champaign
Erik Tjong Kim Sang, University of Groningen
Vivian Tsang, Bloorview Research Institute
Antal van den Bosch, Tilburg University
Aline Villavicencio, Federal University of Rio Grande do Sul
Charles Yang, University of Pennsylvania
Scott Wen-tau Yih, Microsoft Research
Deniz Yuret, Koç University
Luke Zettlemoyer, MIT

Additional Reviewers:

Yee Seng Chan, National University of Singapore
Michael Connor, University of Illinois at Urbana-Champaign
Dmitry Davidov, Hebrew University
Michael Demko, NextJ Systems
Ying Li, Hong Kong University of Science and Technology
Andrew MacKinlay, University of Melbourne
Preslav Nakov, National University of Singapore
Jeremy Nicholson, University of Melbourne
Jing Peng, Montclair State University
Jelena Prokic, University of Groningen
Kevin Small, University of Illinois at Urbana-Champaign
Oren Tsur, Hebrew University

Invited Speakers:

Michael C. Frank, MIT
Andrew McCallum, University of Massachusetts Amherst

Table of Contents

Conference Program

Joint Inference for Natural Language Processing
Andrew McCallum

Modeling Word Learning As Communicative Inference
Michael C. Frank

Thursday, June 4, 2009

9:00–9:15 Opening Remarks

Session 1: Parsing and Tagging

9:15–9:40 *Sample Selection for Statistical Parsers: Cognitively Driven Algorithms and Evaluation Measures*
Roi Reichart and Ari Rappoport

9:40–10:05 *Data-Driven Dependency Parsing of New Languages Using Incomplete and Noisy Training Data*
Kathrin Spreyer and Jonas Kuhn

10:05–10:30 *A Metalearning Approach to Processing the Scope of Negation*
Roser Morante and Walter Daelemans

10:30–11:00 **Coffee Break**

Session 2: Learning Methods

11:00–10:25 *Efficient Linearization of Tree Kernel Functions*
Daniele Pighin and Alessandro Moschitti

11:25–11:40 *A Method for Stopping Active Learning Based on Stabilizing Predictions and the Need for User-Adjustable Stopping*
Michael Bloodgood and Vijay Shanker

11:40–12:40 **Invited Talk:** *Joint Inference for Natural Language Processing*
Andrew McCallum

12:40–14:00 **Lunch**

Thursday, June 4, 2009 (continued)

Shared Task: Overview and Oral Presentations

14:00–14:20 *The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages*
Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue and Yi Zhang

14:20–14:30 *An Iterative Approach for Joint Dependency Parsing and Semantic Role Labeling*
Qifeng Dai, Enhong Chen and Liu Shi

14:30–14:40 *Joint Memory-Based Learning of Syntactic and Semantic Dependencies in Multiple Languages*
Roser Morante, Vincent Van Asch and Antal van den Bosch

14:40–14:50 *Hybrid Multilingual Parsing with HPSG for SRL*
Yi Zhang, Rui Wang and Stephan Oepen

14:50–15:00 *A Latent Variable Model of Synchronous Syntactic-Semantic Parsing for Multiple Languages*
Andrea Gesmundo, James Henderson, Paola Merlo and Ivan Titov

15:00–15:10 *Multilingual Semantic Role Labeling*
Anders Björkelund, Love Hafdell and Pierre Nugues

15:10–15:20 *Multilingual Dependency-based Syntactic and Semantic Parsing*
Wanxiang Che, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin and Ting Liu

15:20–15:30 *Multilingual Dependency Learning: A Huge Feature Engineering Method to Semantic Dependency Parsing*
Hai Zhao, Wenliang Chen, Chunyu Kity and Guodong Zhou

15:30–15:40 *Multilingual Dependency Learning: Exploiting Rich Features for Tagging Syntactic and Semantic Dependencies*
Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto and Kentaro Torisawa

15:40–16:10 **Coffee Break**

Thursday, June 4, 2009 (continued)

Shared Task: Poster Session (16:10–18:00)

Multilingual Semantic Role Labeling
Anders Björkelund, Love Hafdell and Pierre Nugues

Efficient Parsing of Syntactic and Semantic Dependency Structures
Berud Bohnet

Multilingual Dependency-based Syntactic and Semantic Parsing
Wanxiang Che, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin and Ting Liu

An Iterative Approach for Joint Dependency Parsing and Semantic Role Labeling
Qifeng Dai, Enhong Chen and Liu Shi

A Latent Variable Model of Synchronous Syntactic-Semantic Parsing for Multiple Languages
Andrea Gesmundo, James Henderson, Paola Merlo and Ivan Titov

Exploring Multilingual Semantic Role Labeling
Baoli Li, Martin Emms, Saturnino Luz and Carl Vogel

A Second-Order Joint Eisner Model for Syntactic and Semantic Dependency Parsing
Xavier Lluís, Stefan Bott and Lluís Màrquez

Multilingual Semantic Role Labelling with Markov Logic
Ivan Meza-Ruiz and Sebastian Riedel

Joint Memory-Based Learning of Syntactic and Semantic Dependencies in Multiple Languages
Roser Morante, Vincent Van Asch and Antal van den Bosch

The Crotal SRL System : a Generic Tool Based on Tree-structured CRF
Erwan Moreau and Isabelle Tellier

Parsing Syntactic and Semantic Dependencies for Multiple Languages with A Pipeline Approach
Han Ren, Donghong Ji, Jing Wan and Mingyao Zhang

Multilingual Semantic Parsing with a Pipeline of Linear Classifiers
Oscar Täckström

Thursday, June 4, 2009 (continued)

A Joint Syntactic and Semantic Dependency Parsing System based on Maximum Entropy Models
Buzhou Tang, Lu Li, Xinxin Li, Xuan Wang and Xiaolong Wang

Multilingual Syntactic-Semantic Dependency Parsing with Three-Stage Approximate Max-Margin Linear Models
Yotaro Watanabe, Masayuki Asahara and Yuji Matsumoto

A Simple Generative Pipeline Approach to Dependency Parsing and Semantic Role Labeling
Daniel Zeman

Hybrid Multilingual Parsing with HPSG for SRL
Yi Zhang, Rui Wang and Stephan Oepen

Multilingual Dependency Learning: Exploiting Rich Features for Tagging Syntactic and Semantic Dependencies
Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto and Kentaro Torisawa

Multilingual Dependency Learning: A Huge Feature Engineering Method to Semantic Dependency Parsing
Hai Zhao, Wenliang Chen, Chunyu Kity and Guodong Zhou

Friday, June 5, 2009

Session 3: Learning and Semantics

9:00–9:25 *Superior and Efficient Fully Unsupervised Pattern-based Concept Acquisition Using an Unsupervised Parser*
Dmitry Davidov, Roi Reichart and Ari Rappoport

9:25–9:50 *Representing words as regions in vector space*
Katrin Erk

9:50–10:15 *Interactive Feature Space Construction using Semantic Information*
Dan Roth and Kevin Small

10:15–10:40 *Mining the Web for Reciprocal Relationships*
Michael Paul, Roxana Girju and Chen Li

10:40–11:00 **Coffee Break**

Friday, June 5, 2009 (continued)

Session 4: Human Behavior

11:00–11:25 *Minimally Supervised Model of Early Language Acquisition*
Michael Connor, Yael Gertner, Cynthia Fisher and Dan Roth

11:25–11:40 *Learning Where to Look: Modeling Eye Movements in Reading*
Mattias Nilsson and Joakim Nivre

11:40–12:40 **Invited talk:** *Modeling Word Learning As Communicative Inference*
Michael C. Frank

12:40–14:00 **Lunch**

13:45–14:15 **SIGNLL Meeting**

Poster Session (14:00–15:30)

Monte Carlo inference and maximization for phrase-based translation
Abhishek Arun, Chris Dyer, Barry Haddow, Phil Blunsom, Adam Lopez and Philipp Koehn

Investigating Automatic Alignment Methods for Slide Generation from Academic Papers
Brandon Beamer and Roxana Girju

Glen, Glenda or Glendale: Unsupervised and Semi-supervised Learning of English Noun Gender
Shane Bergsma, Dekang Lin and Randy Goebel

Improving Translation Lexicon Induction from Monolingual Corpora via Dependency Contexts and Part-of-Speech Equivalences
Nikesh Garera, Chris Callison-Burch and David Yarowsky

An Intrinsic Stopping Criterion for Committee-Based Active Learning
Fredrik Olsson and Katrin Tomanek

Design Challenges and Misconceptions in Named Entity Recognition
Lev Ratinov and Dan Roth

Automatic Selection of High Quality Parses Created By a Fully Unsupervised Parser
Roi Reichart and Ari Rappoport

Friday, June 5, 2009 (continued)

The NVI Clustering Evaluation Measure
Roi Reichart and Ari Rappoport

Lexical Patterns or Dependency Patterns: Which Is Better for Hypernym Extraction?
Erik Tjong Kim Sang and Katja Hofmann

Improving Text Classification by a Sense Spectrum Approach to Term Expansion
Peter Wittek, Sándor Daránvi and Chew Lim Tan

15:30–16:00 **Coffee Break**

Session 5: Semantic Annotation and Extraction

16:00–16:25 *A simple feature-copying approach for long-distance dependencies*
Marc Vilain, Jonathan Huggins and Ben Wellner

16:25–16:50 *Fine-Grained Classification of Named Entities Exploiting Latent Semantic Kernels*
Claudio Giuliano

16:50–17:15 *Using Encyclopedic Knowledge for Automatic Topic Identification*
Kino Coursey, Rada Mihalcea and William Moen

17:15–17:40 *New Features for FrameNet - WordNet Mapping*
Sara Tonelli and Daniele Pighin

17:40–18:00 Closing remarks and best paper award

Joint Inference for Natural Language Processing

Andrew McCallum
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01002
mccallum@cs.umass.edu

Abstract of the Invited Talk

In recent decades, researchers in natural language processing have made great progress on well-defined subproblems such as part-of-speech tagging, phrase chunking, syntactic parsing, named-entity recognition, coreference and semantic-role labeling. Better models, features, and learning algorithms have allowed systems to perform many of these tasks with 90% accuracy or better. However, success in integrated, end-to-end natural language understanding remains elusive.

I contend that the chief reason for this failure is that errors cascade and accumulate through a pipeline of naively chained components. For example, if we naively use the single most likely output of a part-of-speech tagger as the input to a syntactic parser, and those parse trees as the input to a coreference system, and so on, errors in each step will propagate to later ones: each components 90% accuracy multiplied through six components becomes only 53%.

Consider, for instance, the sentence "I know you like your mother." If a part-of-speech tagger deterministically labels "like" as a verb, then certain later syntactic and semantic analysis will be blocked from alternative interpretations, such as "I know you like your mother (does)." The part-of-speech tagger needs more syntactic and semantic information to make this choice. Consider also the classic example "The boy saw the man with the telescope." No single correct syntactic parse of this sentence is possible in isolation. Correct interpretation requires the integration of these syntactic decisions with semantics and context.

Humans manage and resolve ambiguity by unified, simultaneous consideration of morphology, syntax, semantics, pragmatics and other contextual information. In statistical modeling such unified consideration is known as joint inference. The need for joint inference appears not only in natural language processing, but also in information integration, computer vision, robotics and elsewhere. All of these applications require integrating evidence from multiple sources, at multiple levels of abstraction. I believe that joint inference is one of the most fundamentally central issues in all of artificial intelligence.

In this talk I will describe work in probabilistic models that perform joint inference across multiple components of an information processing pipeline in order to avoid the brittle accumulation of errors. I will survey work in exact inference, variational inference and Markov-chain Monte Carlo methods. We will discuss various approaches that have been applied to natural language processing, and hypothesize about why joint inference has helped in some cases, and not in others.

I will then focus on our recent work at University of Massachusetts in large-scale conditional random fields with complex relational structure. In a single factor graph we seamlessly integrate multiple subproblems, using our new probabilistic programming language to compactly express complex, mutable variable-factor structure both in first-order logic as well as in more expressive Turing-complete imperative procedures. We avoid unrolling this graphical model by using Markov-chain Monte Carlo for inference, and make inference more efficient with learned proposal distributions. Parameter estimation is performed by SampleRank, which avoids complete inference as a subroutine by learning simply to correctly rank successive states of the Markov-chain.

Joint work with Aron Culotta, Michael Wick, Rob Hall, Khashayar Rohanimanesh, Karl Schultz, Sameer Singh, Charles Sutton and David Smith.

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), page 1,
Boulder, Colorado, June 2009.

Modeling word learning as communicative inference

Michael C. Frank
Department of Brain and Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139
mcfrank@mit.edu

Abstract

How do children learn their first words? I describe a model that makes joint inferences about what speakers are trying to talk about and the meanings of the words they use. This model provides a principled framework for integrating a wide variety of non-linguistic information sources into the process of word learning.

Talk Précis

How do children learn their first words? Much work in this field has focused on the social aspects of word learning: that children make use of speakers' intentions—as signaled by a wide range of non-linguistic cues such as their eye-gaze, what they are pointing at, or even what referents are new to them—to infer the meanings of words (Bloom, 2002). However, recent evidence has suggested that adults and children are able to learn words simply from the consistent co-occurrence of words and their referents, even across otherwise ambiguous situations and without explicit social cues as to which referent is being talked about (Yu & Smith 2007; Smith & Yu, 2008).

In this talk I describe work aiming to combine these two sets of evidence within a single probablistic framework (Frank, Goodman, & Tenenbaum, 2009). We propose a model in which learners attempt to infer speakers' moment-to-moment communicative intentions jointly with the meanings of the words they have used to express these intentions. This process of joint inference allows our model to explain away two major sources of noise in simpler statistical word learning proposals: the fact that speakers do not talk about every referent and that not all words that speakers utter are referential.

We find that our model outperforms associative models in learning words accurately from natural corpus data and is able to fit children's behavior in a number of experimental results from developmental psychology. In addition, we have used this basic framework to begin investigating how learners use the rich variety of non-linguistic information signaling speakers' intentions in service of word learning. As an example of this work, I will describe an extension of the model to use discourse continuity as a cue for speakers' intentions.

Acknowledgments

This work supported by a Jacob Javits Graduate Fellowship and NSF Doctoral Dissertation Research Improvement Grant #0746251.

References

Paul Bloom. 2002. *How Children Learn the Meanings of Words*. Cambridge, MA: MIT Press.

Michael C. Frank, Noah D. Goodman, and Joshua B. Tenenbaum. 2009. Using speakers' referential intentions to model early cross-situational word learning. *Psychological Science*.

Linda Smith and Chen Yu. 2008. Infants rapidly learn word-referent mappings via cross-situational statistics. *Cognition*, 106, 1558-1568.

Chen Yu and Linda Smith. 2007. Rapid word learning under uncertainty via cross-situational statistics. *Psychological Science*, 18, 414-420.

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), page 2,
Boulder, Colorado, June 2009.

Sample Selection for Statistical Parsers: Cognitively Driven Algorithms and Evaluation Measures

Roi Reichart
ICNC
Hebrew University of Jerusalem
roiri@cs.huji.ac.il

Ari Rappoport
Institute of computer science
Hebrew University of Jerusalem
arir@cs.huji.ac.il

Abstract

Creating large amounts of manually annotated training data for statistical parsers imposes heavy cognitive load on the human annotator and is thus costly and error prone. It is hence of high importance to decrease the human efforts involved in creating training data without harming parser performance. For constituency parsers, these efforts are traditionally evaluated using the total number of constituents (TC) measure, assuming uniform cost for each annotated item. In this paper, we introduce novel measures that quantify aspects of the cognitive efforts of the human annotator that are not reflected by the TC measure, and show that they are well established in the psycholinguistic literature. We present a novel *parameter based sample selection* approach for creating good samples in terms of these measures. We describe methods for global optimisation of lexical parameters of the sample based on a novel optimisation problem, the *constrained multiset multicover* problem, and for *cluster-based sampling* according to syntactic parameters. Our methods outperform previously suggested methods in terms of the new measures, while maintaining similar TC performance.

1 Introduction

State of the art statistical parsers require large amounts of manually annotated data to achieve good performance. Creating such data imposes heavy cognitive load on the human annotator and is thus costly and error prone. Statistical parsers are major components in NLP applications such as QA (Kwok et al., 2001), MT (Marcu et al., 2006) and SRL (Toutanova et al., 2005). These often operate over the highly variable Web, which consists of texts written in many languages and genres. Since the performance of parsers markedly degrades when training and test data come from different domains (Lease and Charniak, 2005), large amounts of training data from each domain are required for using them effectively. Thus, decreasing the human efforts involved in creating training data for parsers without harming their performance is of high importance.

In this paper we address this problem through *sample selection*: given a parsing algorithm and a large pool of unannotated sentences S, select a subset $S_1 \subset S$ for human annotation such that the human efforts in annotating S_1 are minimized while the parser performance when trained with this sample is maximized.

Previous works addressing training sample size vs. parser performance for constituency parsers (Section 2) evaluated training sample size using the total number of constituents (TC). Sentences differ in length and therefore in annotation efforts, and it has been argued (see, e.g, (Hwa, 2004)) that TC reflects the number of decisions the human annotator makes when syntactically annotating the sample, assuming uniform cost for each decision.

In this paper we posit that important aspects of the efforts involved in annotating a sample are not reflected by the TC measure. Since annotators analyze sentences rather than a bag of constituents, sentence structure has a major impact on their cognitive efforts. Sizeable psycholinguistic literature points to the connection between nested structures in the syntactic structure of a sentence and its annotation efforts. This has motivated us to introduce (Section 3) three sample size measures, the total and av-

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 3–11,
Boulder, Colorado, June 2009.

erage number of nested structures of degree k in the sample, and the average number of constituents per sentence in the sample.

Active learning algorithms for sample selection focus on sentences that are difficult for the parsing algorithm when trained with the available training data (Section 2). In Section 5 we show that active learning samples contain a high number of complex structures, much higher than their number in a randomly selected sample that achieves the same parser performance level. To avoid that, we introduce (Section 4) a novel *parameter based sample selection* (PBS) approach which aims to select a sample that enables good estimation of the model parameters, without focusing on difficult sentences. In Section 5 we show that the methods derived from our approach select substantially fewer complex structures than active learning methods and the random baseline.

We propose two different methods. In *cluster based sampling* (CBS), we aim to select a sample in which the distribution of the model parameters is similar to their distribution in the whole unlabelled pool. To do that we build a vector representation for each sentence in the unlabelled pool reflecting the distribution of the model parameters in this sentence, and use a clustering algorithm to divide these vectors into clusters. In the second method we use the fact that a sample containing many examples of a certain parameter yields better estimation of this parameter. If this parameter is crucial for model performance and the selection process does not harm the distribution of other parameters, then the selected sample is of high quality. To select such a sample we introduce a reduction between this selection problem and a variant of the NP-hard multiset-multicover problem (Hochbaum, 1997). We call this problem the *constrained multiset multicover* (CMM) problem, and present an algorithm to approximate it.

We experiment (Section 5) with the WSJ PennTreebank (Marcus et al., 1994) and Collins' generative parser (Collins, 1999), as in previous work. We show that PBS algorithms achieve good results in terms of both the traditional TC measure (significantly better than the random selection baseline and similar to the results of the state of the art tree entropy (TE) method of (Hwa, 2004)) and our novel cognitively driven measures (where PBS algorithms significantly outperform both TE and the random baseline). We thus argue that PBS provides a way to select a sample that imposes reduced cognitive load on the human annotator.

2 Related Work

Previous work on sample selection for statistical parsers applied active learning (AL) (Cohn and Ladner, 1994) to corpora of various languages and syntactic annotation schemes and to parsers of different performance levels. In order to be able to compare our results to previous work targeting high parser performance, we selected the corpus and parser used by the method reporting the best results (Hwa, 2004), WSJ and Collins' parser.

Hwa (2004) used uncertainty sampling with the tree entropy (TE) selection function[1] to select training samples for the Collins parser. In each iteration, each of the unlabelled pool sentences is parsed by the parsing model, which outputs a list of trees ranked by their probabilities. The scored list is treated as a random variable and the sentences whose variable has the highest entropy are selected for human annotation. Sample size was measured in TC and ranged from 100K to 700K WSJ constituents. The initial size of the unlabelled pool was 800K constituents (the 40K sentences of sections 2-21 of WSJ). A detailed comparison between the results of TE and our methods is given in Section 5.

The following works addressed the task of sample selection for statistical parsers, but in significantly different experimental setups. Becker and Osborne (2005) addressed lower performance levels of the Collins parser. Their uncertainty sampling protocol combined bagging with the TE function, achieving a 32% TC reduction for reaching a parser f-score level of 85.5%. The target sample size set contained a much smaller number of sentences ($\sim$5K) than ours. Baldridge and Osborne (2004) addressed HPSG parse selection using a feature based log-linear parser, the Redwoods corpus and committee based active learning, obtaining 80% reduction in annotation cost. Their annotation cost measure was related to the number of possible parses of the sentence. Tang et al. (2002) addressed a shallow parser trained on a semantically annotated corpus.

[1] Hwa explored several functions in the experimental setup used in the present work, and TE gave the best results.

They used an uncertainty sampling protocol, where in each iteration the sentences of the unlabelled pool are clustered using a distance measure defined on parse trees to a predefined number of clusters. The most uncertain sentences are selected from the clusters, the training taking into account the densities of the clusters. They reduced the number of training sentences required for their parser to achieve its best performance from 1300 to 400.

The importance of cognitively driven measures of sentences' syntactic complexity has been recognized by Roark et al. (2007) who demonstrated their utility for mild cognitive impairment diagnosis. Zhu et al. (2008) used a clustering algorithm for sampling the initial labeled set in an AL algorithm for word sense disambiguation and text classification. In contrast to our CBS method, they proceeded with iterative uncertainty AL selection. Melville et al. (2005) used parameter-based sample selection for a classifier in a classic active learning setting, for a task very different from ours.

Sample selection has been applied to many NLP applications. Examples include base noun phrase chunking (Ngai, 2000), named entity recognition (Tomanek et al., 2007) and multi–task annotation (Reichart et al., 2008).

3 Cognitively Driven Evaluation Measures

While the resources, capabilities and constraints of the human parser have been the subject of extensive research, different theories predict different aspects of its observed performance. We focus on structures that are widely agreed to impose a high cognitive load on the human annotator and on theories considering the cognitive resources required in parsing a complete sentence. Based on these, we derive measures for the cognitive load on the human parser when syntactically annotating a set of sentences.

Nested structures. A nested structure is a parse tree node representing a constituent created while another constituent is still being processed ('open'). The *degree* K of a nested structure is the number of such open constituents. In this paper, we enumerate the constituents in a top-down left-right order, and thus when a constituent is created, only its ancestors are processed[2]. A constituent is processed until the processing of its children is completed. For example, in Figure 1, when the constituent NP3 is created, it starts a nested structure of degree 2, since two levels of its ancestors (VP, S) are still processed. Its parent (VP) starts a nested structure of degree 1.

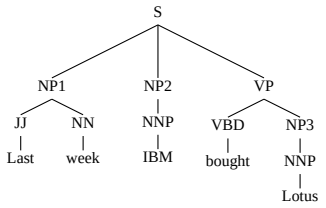


Figure 1: An example parse tree.

The difficulty of deeply nested structures for the human parser is well established in the psycholinguistics literature. We review here some of the various explanations of this phenomenon; for a comprehensive review see (Gibson, 1998).

According to the classical stack overflow theory (Chomsky and Miller, 1963) and its extension, the incomplete syntactic/thematic dependencies theory (Gibson, 1991), the human parser should track the open structures in its short term memory. When the number of these structures is too large or when the structures are nested too deeply, the short term memory fails to hold them and the sentence becomes uninterpretable.

According to the perspective shifts theory (MacWhinney, 1982), processing deeply nested structures requires multiple shifts of the annotator perspective and is thus more difficult than processing shallow structures. The difficulty of deeply nested structured has been demonstrated for many languages (Gibson, 1998).

We thus propose the total number of nested structures of degree K in a sample (TNSK) as a measure of the cognitive efforts that its annotation requires. The higher K is, the more demanding the structure.

Sentence level resources. In the psycholinguistic literature of sentence processing there are many theories describing the cognitive resources required during a complete sentence processing. These resources might be allocated during the processing of a certain word and are needed long after its constituent is closed. We briefly discuss two lines of theory, focusing on their predictions that sentences consisting of a large number of structures (e.g., con-

[2] A good review on node enumeration of the human parser in given in (Abney and Johnson, 1991).

stituents or nested structures) require more cognitive resources for longer periods.

Levelt (2001) suggested a layered model of the mental lexicon organization, arguing that when one hears or reads a sentence s/he activates word forms (lexemes) that in turn activate lemma information. The lemma information contains information about syntactic properties of the word (e.g., whether it is a noun or a verb) and about the possible sentence structures that can be generated given that word. The process of reading words and retrieving their lemma information is incremental and the lemma information for a given word is used until its syntactic structure is completed. The information about a word include all syntactic predictions, obligatory (e.g., the prediction of a noun following a determiner) and optional (e.g., optional arguments of a verb, modifier relationships). This information might be relevant long after the constituents containing the word are closed, sometimes till the end of the sentence.

Another line of research focuses on working memory, emphasizing the *activation decay* principle. It stresses that words and structures perceived during sentence processing are forgotten over time. As the distance between two related structures in a sentence grows, it is more demanding to reactivate one when seeing the other. Indeed, supported by a variety of observations, many of the theories of the human parser (see (Lewis et al., 2006) for a survey) predict that processing items towards the end of longer sentences should be harder, since they most often have to be integrated with items further back. Thus, sentences with a large number of structures impose a special cognitive load on the annotator.

We thus propose to use the number of structures (constituents or nested structures) in a sentence as a measure of its difficulty for human annotation. The measures we use for a sample (a sentence set) are the *average number of constituents* (AC) and the *average number of nested structures of degree* k (ANSK) per sentence in the set. Higher AC or ANSK values of a set imply higher annotation requirements[3].

Pschycolinguistics research makes finer observations about the human parser than those described here. A complete survey of that literature is beyond the scope of this paper. We consider the proposed measures a good approximation of some of the human parser characteristics.

[3]The correlation between the number of constituents and sentence length is very strong (e.g., correlation coefficient of 0.93 in WSJ section 0). We could use the number of words, but we prefer the number of structures since the latter better reflects the arguments made in the literature.

4 Parameter Based Sampling (PBS)

Our approach is to sample the unannotated pool with respect to the distribution of the model parameters in its sentences. In this paper, in order to compare to previous works, we apply our methods to the Collins generative parser (Collins, 1999). For any sentence s and parse tree t it assigns a probability $p(s,t)$, and finds the tree for which this probability is maximized. To do that, it writes $p(t,s)$ as a product of the probabilities of the constituents in t and decomposes the latter using the chain rule. In simplified notation, it uses:

$$p(t,s) = \prod P(S_1 \rightarrow S_2 \dots S_n) = \prod P(S_1) \cdot \dots \cdot P(S_n|\phi(S_1 \dots S_n)) \quad (1)$$

We refer to the conditional probabilities as the model *parameters*.

Cluster Based Sampling (CBS). We describe here a method for sampling subsets that leads to a parameter estimation that is similar to the parameter estimation we would get if annotating the whole unannotated set.

To do that, we randomly select M sentences from the unlabelled pool N, manually annotate them, train the parser with these sentences and parse the rest of the unlabelled pool ($G = N - M$). Using this annotation we build a syntactic vector representation for each sentence in G. We then cluster these sentences and sample the clusters with respect to their weights to preserve the distribution of the syntactic features. The selected sentences are manually annotated and combined with the group of M sentences to train the final parser. The size of this combined sample is measured when the annotation efforts are evaluated.

Denote the left hand side nonterminal of a constituent by P and the unlexicalized head of the constituent by H. The domain of P is the set of nonterminals (excluding POS tags) and the domain of H is the set of nonterminals and POS tags of WSJ. In all the parameters of the Collins parser P and H are conditioned upon. We thus use (P,H) pairs as the

features in the vector representation of each sentence in G. The i-th coordinate is given by the equation:

$$\sum_{c \in t(s)} \sum_{i} F_i(Q(c) == i) \cdot L(c) \quad (2)$$

Where c are the constituents of the sentence parse $t(s)$, Q is a function that returns the (P, H) pair of the constituent c, F_i is a predicate that returns 1 iff it is given pair number i as an argument and 0 otherwise, and L is the number of modifying non-terminals in the constituent plus 1 (for the head), counting the number of parameters that condition on (P, H). Following equation (2), the ith coordinate of the vector representation of a sentence in G contains the number of parameters that will be calculated conditioned on the ith (P, H) pair.

We use the k-means clustering algorithm, with the L_2 norm as a distance metric (MacKay, 2002), to divide vectors into clusters. Clusters created by this algorithm contain adjacent vectors in a Euclidean space. Clusters represent sentences with similar features values. To initialize k-means, we sample the initial centers values from a uniform distribution over the data points.

We do not decide on the number of clusters in advance but try to find inherent structure in the data. Several methods for estimating the 'correct' number of clusters are known (Milligan and Cooper, 1985). We used a statistical heuristic called the elbow test. We define the 'within cluster dispersion' W_k as follows. Suppose that the data is divided into k clusters $C_1 \dots C_k$ with $|C_j|$ points in the jth cluster. Let $D_t = \sum_{i,j \in C_t} d_{i,j}$ where $d_{i,j}$ is the squared Euclidean distance, then $W_k := \sum_{t=1}^{k} \frac{1}{2|C_t|} D_t$. W_k tends to decrease monotonically as k increases. In many cases, from some k this decrease flattens markedly. The heuristic is that the location of such an 'elbow' indicates the appropriate number of clusters. In our experiments, an obvious elbow occurred for 15 clusters.

k_i sentences are randomly sampled from each cluster, $k_i = D \frac{|C_i|}{\sum_j |C_j|}$, where D is the number of sentences to be sampled from G. That way we ensure that in the final sample each cluster is represented according to its size.

CMM Sampling. All of the parameters in the Collins parser are conditioned on the constituent's head word. Since word statistics are sparse, sampling from clusters created according to a lexical vector representation of the sentences does not seem promising[4].

Another way to create a sample from which the parser can extract robust head word statistics is to select a sample containing many examples of each word. More formally, we denote the words that occur in the unlabelled pool at least t times by *t-words*, where t is a parameter of the algorithm. We want to select a sample containing at least t examples of as many t-words as possible.

To select such a sample we introduce a novel optimisation problem. Our problem is a variant of the multiset multicover (MM) problem, which we call the *constrained multiset multicover* (CMM) problem. The setting of the MM problem is as follows (Hochbaum, 1997): Given a set I of m elements to be covered each b_i times, a collection of multisets $S_j \subset I$, $j \in J = \{1, \dots, n\}$ (a multiset is a set in which members' multiplicity may be greater than 1), and weights w_j, find a subcollection C of multisets that covers each $i \in I$ at least b_i times, and such that $\sum_{j \in C} w_j$ is minimized.

CMM differs from MM in that in CMM the sum of the weights (representing the desired number of sentences to annotate) is bounded, while the number of covered elements (representing the t-words) should be maximized. In our case, I is the set of words that occur at least t times in the unlabelled pool, $b_i = t, \forall i \in I$, the multisets are the sentences in that pool and $w_j = 1, \forall j \in J$.

Multiset multicover is NP-hard. However, there is a good greedy approximation algorithm for it. Define $a(s_j, i) = min(R(s_j, i), d_i)$, where d_i is the difference between b_i and the number of instances of item i that are present in our current sample, and $R(s_j, i)$ is the multiplicity of the i-th element in the multiset s_j. Define $A(s_j)$ to be the multiset containing exactly $a(s_j, i)$ copies of any element i if s_j is not already in the set cover and the empty set if it is. The greedy algorithm repeatedly adds a set minimizing $\frac{w_j}{|A(s_j)|}$. This algorithm provenly achieves an approximation ratio between $ln(m)$ and $ln(m) + 1$. In our case all weights are 1, so the algorithm would

[4] We explored CBS with several lexical features schemes and got only marginal improvement over random selection.

simply add the sentence that maximizes $A(s_j)$ to the set cover.

The problem in directly applying the algorithm to our case is that it does not take into account the desired sample size. We devised a variant of the algorithm where we use a binary tree to 'push' upwards the number of t-words in the whole batch of unannotated sentences that occurs at least t times in the selected one. Below is a detailed description. D denotes the desired number of items to sample.

The algorithm has two steps. First, we iteratively sample (without replacement) D multisets (sentences) from a uniform distribution over the multisets. In each iteration we calculate for the selected multiset its 'contribution' – the number of items that cross the threshold of t occurrences with this multiset minus the number of items that cross the t threshold without this multiset (i.e. the contribution of the first multiset is the number of t-words occurring more than t times in it). For each multiset we build a node with a key that holds its contribution, and insert these nodes in a binary tree. Insertion is done such that all downward paths are sorted in decreasing order of key values.

Second, we iteratively sample (from a uniform distribution, without replacement) the rest of the multisets pool. For each multiset we perform two steps. First, we prepare a node with a key as described above. We then randomly choose Z leaves[5] in the binary tree (if the number of leaves is smaller than Z all of the leaves are chosen). For each leaf we find the place of the new node in the path from the root to the leaf (paths are sorted in decreasing order of key values). We insert the new node to the highest such place found (if the new key is not smaller than the existing paths), add its multiset to the set of selected multisets, and remove the multiset that corresponds to the leaf of this path from the batch and the leaf itself from the binary tree. We finally choose the multisets that correspond to the highest D nodes in the tree.

An empirical demonstration of the quality of approximation that the algorithm provides is given in Figure 2. We ran our algorithm with the threshold parameter set to $t \in [2, 14]$ and counted the number of words occurring at least 5 times in the selected sample. We followed the same experimental protocol as in Section 5. The graph shows that the number of words occurring at least 5 times in a sample selected by our algorithm when $t = 5$ is significantly higher (by about a 1000) than the number of such words in a randomly selected sample and in samples selected by our algorithm with other t parameter values. We got the same pattern of results when counting words occurring at least t times for the other values of the t parameter – only the run of the algorithm with the corresponding t value created a sample with significantly higher number of words not below threshold. The other runs and random selection resulted in samples containing significantly lower number of words not below threshold.

In Section 5 we show that the parser performance when it is trained with a sample selected by CMM is significantly better than when it is trained with a randomly selected sample. Improvement is similar across the t parameter values.

[5]We tried Z values from 10 to 100 in steps of 10 and observed very similar results. We report results for $Z = 100$.

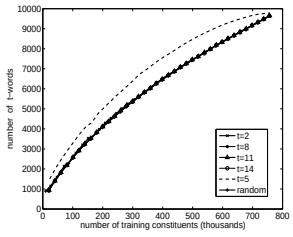


Figure 2: Number of t-words for t = 5 in samples selected by CMM runs with different values of the threshold parameter t and in a randomly selected sample. CMM with t = 5 is significantly higher. All the lines except for the line for t = 5 are unified. For clarity, we do not show all t values: their curves are also similar to the $t \neq 5$ lines.

Method	**86%**	**86.5%**	**87%**	**87.5%**	**88%**
TE	16.9% (152K)	**27.1% (183K)**	**26.9% (258K)**	14.8% (414K)	**15.8% (563 K)**
CBS	**19.6% (147K)**	16.8% (210K)	19% (286K)	**21.1% (382K)**	9% (610K)
CMM	9% (167K)	10.4% (226K)	8.9% (312K)	10.3% (433K)	14% (574K)

Table 1: Reduction in annotation cost in TC terms compared to the random baseline for tree entropy (TE), syntactic clustering (CBS) and CMM. The compared samples are the smallest samples selected by each of the methods that achieve certain f-score levels. Reduction is calculated by: $100 - 100 \times (TC_{\text{method}}/TC_{\text{random}})$.

	86%				87%				88%			
Method	TNSK (1-6)	TNSK (7-22)	ANSK (1-6)	ANSK (7-22)	TNSK (1-6)	TNSK (7-22)	ANSK (1-6)	ANSK (7-22)	TNSK (1-6)	TNSK (7-22)	ANSK (1-6)	ANSK (7-22)
TE	**34.9**%	3.6%	- 8.9%	- 61.3%	**42.2**%	14.4%	- 9.9%	- 62.7%	**25**%	8.1%	- 6.3%	- 30%
CBS	21.3%	**18.6**%	- **0.5**%	- 3.5%	19.6%	**24.2**%	- **0.3**%	- **1.8**%	8.9%	8.6 %	**0**%	- 0.3%
CMM	10.18%	8.87%	-0.82%	-**3.39%**	11%	16.22%	-0.34%	-**1.8%**	14.65%	**14.11**%	-0.02%	- **0.08**%

Table 2: Annotation cost reduction in TNSK and ANSK compared to the random baseline for tree entropy (TE), syntactic clustering (CBS) and CMM. The compared samples are the smallest selected by each of the methods that achieve certain f-score levels. Each column represents the reduction in total or average number of structures of degree 1–6 or 7–22. Reduction for each measure is calculated by: $100 - 100 \times (measure_{\text{method}}/measure_{\text{random}})$. Negative reduction is an addition. Samples with a higher reduction in a certain measure are better in terms of that measure.

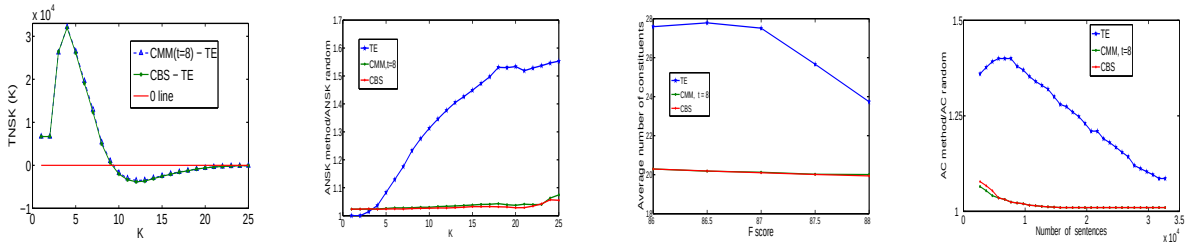


Figure 3: Left to right: First: The difference between the number of nested structures of degree K of CMM and TE and of CBS and TE. The curves are unified. The 0 curve is given for reference. Samples selected by CMM and CBS have more nested structures of degrees 1–6 and less nested structures of degrees 7–22. Results are presented for the smallest samples required for achieving f-score of 88. Similar patterns are observed for other f-score values. Second: Average number of nested structures of degree K as a function of K for the smallest sample required for achieving f-score of 88. Results for each of the methods are normalized by the average number of nested structures of degree K in the smallest randomly selected sample required for achieving f-score of 88. The sentences in CMM and CBS samples are not more complex than sentences in a randomly selected sample. In TE samples sentences are more complex. Third: Average number of constituents (AC) for the smallest sample of each of the methods that is required for achieving a given f-score. CMM and CBS samples contain sentences with a smaller number of constituents. Fourth: AC values for the samples created by the methods (normalized by AC values of a randomly selected sample). The sentences in TE samples, but not in CMM and CBS samples, are more complex than sentences in a randomly selected sample.

5 Results

Experimental setup. We used Bikel's reimplementation of Collins' parsing model 2 (Bikel, 2004). Sections 02-21 and 23 of the WSJ were stripped from their annotation. Sections 2-21 (39832 sentences, about 800K constituents) were used for training, Section 23 (2416 sentences) for testing. No development set was used. We used the gold standard POS tags in two cases: in the test section (23) in all experiments, and in Sections 02-21 in the CBS method when these sections are to be parsed in the process of vector creation. In active learning methods the unlabelled pool is parsed in each iteration and thus should be tagged with POS tags. Hwa (2004) (to whom we compare our results) used the gold standard POS tags for the same sections in her work[6]. We implemented a random baseline where sentences are uniformly selected from the unlabelled pool for annotation. For reliability we repeated each experiment with the algorithms and the random baseline 10 times, each time with different random selections (M sentences for creating syntactic tagging and k-means initialization for CBS, sentence order in CMM), and averaged the results.

Each experiment contained 38 runs. In each run a different desired sample size was selected, from 1700 onwards, in steps of 1000. Parsing performance is measured in terms of f-score

Results. We compare the performance of our CBS and CMM algorithms to the TE method (Hwa, 2004)[7], which is the only sample selection work ad-

[6]Personal communication with Hwa. Collins' parser uses an input POS tag only if it cannot tag its word using the statistics learned from the training set.

[7]Hwa has kindly sent us the samples selected by her TE. We evaluated these samples with TC and the new measures. The TC of the minimal sample she sent us needed for achieving f-score

dressing our experimental setup. Unless otherwise stated, we report the reduction in annotation cost: $100 - 100 \times (measure_{\text{method}}/measure_{\text{random}})$. CMM results are very similar for $t \in \{2, 3, \ldots, 14\}$, and presented for $t = 8$.

Table 1 presents reduction in annotation cost in TC terms. CBS achieves greater reduction for $f = 86, 87.5$, TE for $f = 86.5, 87, 88$. For $f = 88$, TE and CMM performance are almost similar. Examining the f-score vs. TC sample size over the whole constituents range (not shown due to space constraints) reveals that CBS, CMM and TE outperform random selection over the whole range. CBS and TE performance are quite similar with TE being better in the ranges of 170–300K and 520–650K constituents (42% of the 620K constituents compared) and CBS being better in the ranges of 130–170K and 300–520K constituents (44% of the range). CMM performance is worse than CBS and TE until 540K constituents. From 650K constituents on, where the parser achieves its best performance, the performance of CMM and TE methods are similar, outperforming CBS.

Table 2 shows the annotation cost reduction in ANSK and TNSK terms. TE achieves remarkable reduction in the total number of relatively shallow structures (TNSK K = 1–6). Our methods, in contrast, achieve remarkable reduction in the number of deep structures (TNSK K = 7–22)[8]. This is true for all f-score values. Moreover, the average number of nested structures per sentence, for every degree K (ANSK for every K) in TE sentences is much higher than in sentences of a randomly selected sample. In samples selected by our methods, the ANSK values are very close to the ANSK values of randomly selected samples. Thus, sentences in TE samples are much more complex than in CBS and CMM samples.

The two leftmost graphs in Figure 3 demonstrate (for the minimal samples required for f-score of 88) that these reductions hold for each K value (ANSK) and for each $K \in [7, 22]$ (TNSK) not just on the average over these K values. We observed similar results for other f-score values.

The two rightmost graphs of Figure 3 demonstrates AC results. The left of them shows that for every f-score value, the AC measure of the minimal TE sample required to achieve that f-score is higher than the AC value of PBS samples (which are very similar to the AC values of randomly selected samples). The right graph demonstrates that for every sample size, the AC value of TE samples is higher than that of PBS samples.

All AL based previous work (including TE) is iterative. In each iteration thousands of sentences are parsed, while PBS algorithms perform a single iteration. Consequently, PBS computational complexity is dramatically lower. Empirically, using a Pentium 4 2.4GHz machine, CMM requires about an hour and CBS about 16.5 hours, while the TE parsing steps alone take 662 hours (27.58 days).

6 Discussion and Future Work

We introduced novel evaluation measures: AC, TNSK and ANSK for the task of sample selection for statistical parsers. Based on the psycholinguistic literature we argue that these measures reflect aspects of the cognitive efforts of the human annotator that are not reflected by the traditional TC measure. We introduced the parameter based sample selection (PBS) approach and its CMM and CBS algorithms that do not deliberately select difficult sentences. Therefore, our intuition was that they should select a sample that leads to an accurate parameter estimation but does not contain a high number of complex structures. We demonstrated that CMM and CBS achieve results that are similar to the state of the art TE method in TC terms and outperform it when the cognitively driven measures are considered.

The measures we suggest do not provide a full and accurate description of human annotator efforts. In future work we intend to extend and refine our measures and to revise our algorithms accordingly.

We also intend to design stopping criteria for the PBS methods. These are criteria that decide when the selected sample suffices for the parser best performance and further annotation is not needed.

of 88 is different from the number reported in (Hwa, 2004). We compare our TC results with the TC result in the sample sent us by Hwa.

[8] We present results where the border between shallow and deep structures is set to be $K_{border} = 6$. For every $K_{border} \in \{7, 8, \ldots, 22\}$ TNSK reductions with CBS and CMM are much more impressive than with TE for structures whose degree is $K \in [K_{border}, 22]$.

References

Steven Abney and Mark Johnson, 1991. Memory requirements and local ambiguities of parsing strategies. *Psycholinguistic Research*, 20(3):233–250.

Daniel M. Biken, 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Jason Baldridge and Miles Osborne, 2004. Active learning and the total cost of annotation. *EMNLP '04*.

Markus Becker and Miles Osborne, 2005. A two-stage method for active learning of statistical grammars. *IJCAI 05*.

Markus Becker, 2008. Active learning – an explicit treatment of unreliable parameters. Ph.D. thesis, The University of Edinburgh.

Noam Chomsky and George A. Miller, 1963. Finitary models of language users. In R. Duncan Luce, Robert R. Bush, and Eugene Galanter, editors, *Handbook of Mathematical Psychology*, volume II. John Wiley, New York, 419–491.

David Cohn, Les Atlas and Richard E. Ladner, 1994. Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

Michael Collins, 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Edward Gibson, 1991. *A computational theory of human linguistic processing: memory limitations and processing breakdowns.* Ph.D. thesis, Carnegie Mellon University, Pittsburg, PA.

Edward Gibson, 1998. Linguistic complexity: locality of syntactic dependencies. *Cognition*, 68:1–76.

Dorit Hochbaum (ed), 1997. *Approximation algorithms for NP-hard problems*. PWS Publishing, Boston.

Rebecca Hwa, 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276.

Cody Kwok, Oren Etzioni, Daniel S. Weld, 2001. Scaling question answering to the Web. *WWW '01*.

Matthew Lease and Eugene Charniak, 2005. Parsing biomedical literature. *IJCNLP '05*.

Willem J.M. Levelt, 2001. Spoken word production: A theory of lexical access. *PNAS*, 98(23):13464–13471.

Richard L. Lewis, Shravan Vasishth and Julie Van Dyke, 2006. Computational principles of working memory in sentence comprehension. *Trends in Cognitive Science*, October:447-454.

David MacKay, 2002. *Information theory, inference and learning algorithms*. Cambridge University Press.

Brian MacWhinney, 1982. The competition model. In B. MacWhinney, editor, *Mechanisms of language acquisition*. Hillsdale, NJ: Lawrence Erlbaum, 249–308.

Daniel Marcu, Wei Wang, Abdessamabad Echihabi, and Kevin Knight, 2006. SPMT: Statistical machine translation with syntactified target language phrases. *EMNLP '06.*

P. Melville, M. Saar-Tsechansky, F. Provost and R.J. Mooney, 2005. An expected utility approach to active feature-value acquisition. *5th IEEE Intl. Conf. on Data Mining '05.*

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz, 1994. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

G.W. Milligan and M.C Cooper, 1985. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 58(2):159–157.

Grace Ngai and David Yarowski, 2000. Rule writing or annotation: cost–efficient resource usage for base noun phrase chunking. *ACL '00*.

Roi Reichart, Katrin Tomanek, Udo Hahn and Ari Rappoport, 2008. Multi-task active learning for linguistic annotations. *ACL '08.*

Brian Roark, Margaret Mitchell and Kristy Hollingshead, 2007. Syntactic complexity measures for detecting mild cognitive impairment. *BioNLP workshop, ACL '07*.

Min Tang, Xiaoqiang Luo, and Salim Roukos, 2002. Active learning for statistical natural language parsing. *ACL '02.*

Katrin Tomanek, Joachim Wermtre, and Udo Hahn, 2007. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. *EMNLP '07.*

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning, 2005. Joint learning improves semantic role labeling. *ACL '05*.

Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou, 2008. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. *COLING '08.*

Data-Driven Dependency Parsing of New Languages Using Incomplete and Noisy Training Data

Kathrin Spreyer and **Jonas Kuhn**
Department of Linguistics
University of Potsdam, Germany
{spreyer,kuhn}@ling.uni-potsdam.de

Abstract

We present a simple but very effective approach to identifying high-quality data in noisy data sets for structured problems like parsing, by greedily exploiting partial structures. We analyze our approach in an annotation projection framework for dependency trees, and show how dependency parsers from two different paradigms (graph-based and transition-based) can be trained on the resulting tree fragments. We train parsers for Dutch to evaluate our method and to investigate to which degree graph-based and transition-based parsers can benefit from incomplete training data. We find that partial correspondence projection gives rise to parsers that outperform parsers trained on aggressively filtered data sets, and achieve unlabeled attachment scores that are only 5% behind the average UAS for Dutch in the CoNLL-X Shared Task on supervised parsing (Buchholz and Marsi, 2006).

1 Introduction

Many weakly supervised approaches to NLP rely on heuristics or filtering techniques to deal with noise in unlabeled or automatically labeled training data, e.g., in the exploitation of parallel corpora for cross-lingual projection of morphological, syntactic or semantic information. While heuristic approaches can implement (linguistic) knowledge that helps to detect noisy data (e.g., Hwa et al. (2005)), they are typically task- and language-specific and thus introduce a component of indirect supervision. Non-heuristic filtering techniques, on the other hand, employ reliability measures (often unrelated to the task) to predict high-precision data points (e.g., Yarowsky et al. (2001)). In order to reach a sufficient level of precision, filtering typically has to be aggressive, especially for highly structured tasks like parsing. Such aggressive filtering techniques incur massive data loss and enforce trade-offs between the quality and the amount of usable data.

Ideally, a general filtering strategy for weakly supervised training of structured analysis tools should eliminate noisy subparts in the automatic annotation without discarding its high-precision aspects; thereby data loss would be kept to a minimum. In this paper, we propose an extremely simple approach to noise reduction which greedily exploits partial correspondences in a parallel corpus, i.e., correspondences potentially covering only substructures of translated sentences. We implemented this method in an annotation projection framework to create training data for two dependency parsers representing different parsing paradigms: The MSTParser (McDonald et al., 2005) as an instance of *graph-based dependency parsing*, and the MaltParser (Nivre et al., 2006) to represent *transition-based dependency parsing*. In an empirical evaluation, we investigate how they react differently to incomplete and noisy training data.

Despite its simplicity, the partial correspondence approach proves very effective and leads to parsers that achieve unlabeled attachment scores that are only 5% behind the average UAS for Dutch in the CoNLL-X Shared Task (Buchholz and Marsi, 2006).

After a summary of related work in Sec. 2, we discuss dependency tree projection (Sec. 3) and partial correspondence (Sec. 4). In Sec. 5, we give an overview of graph- and transition-based dependency parsing and describe how each can be adapted for training on partial training data in Sec. 6. Experimental results are presented in Sec. 7, followed by an analysis in Sec. 8. Sec. 9 concludes.

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 12–20,
Boulder, Colorado, June 2009.

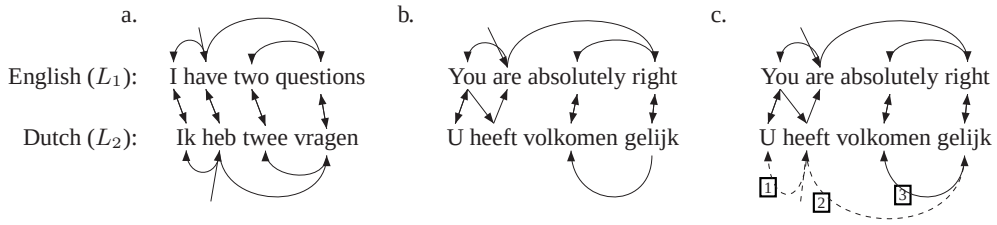


Figure 1: Dependency tree projection from English to Dutch. (a) Ideal scenario with bidirectional alignments. (b) Projection fails due to weak alignments. (c) Constrained fallback projection.

2 Related Work

Annotation projection has been applied to many different NLP tasks. On the word or phrase level, these include morphological analysis, part-of-speech tagging and NP-bracketing (Yarowsky et al., 2001), temporal analysis (Spreyer and Frank, 2008), or semantic role labeling (Padó and Lapata, 2006). In these tasks, word labels can technically be introduced in isolation, without reference to the rest of the annotation. This means that an aggressive filter can be used to discard unreliable data points (words in a sentence) without necessarily affecting high-precision data points in the same sentence. By using only the bidirectional word alignment links, one can implement a very robust such filter, as the bidirectional links are generally reliable, even though they have low recall for overall translational correspondences (Koehn et al., 2003). The bidirectional alignment filter is common practice (Padó and Lapata, 2006); a similar strategy is to discard entire sentences with low aggregated alignment scores (Yarowsky et al., 2001).

On the sentence level, Hwa et al. (2005) were the first to project dependency trees from English to Spanish and Chinese. They identify unreliable target parses (as a whole) on the basis of the number of unaligned or over-aligned words. In addition, they manipulate the trees to accommodate for non-isomorphic sentences. Systematic non-parallelisms between source and target language are then addressed by hand-crafted rules in a post-projection step. These rules account for an enormous increase in the unlabeled f-score of the direct projections, from 33.9 to 65.7 for Spanish and from 26.3 to 52.4 for Chinese. But they need to be designed anew for every target language, which is time-consuming and requires knowledge of that language.

Research in the field of unsupervised and weakly supervised parsing ranges from various forms of EM training (Pereira and Schabes, 1992; Klein and Manning, 2004; Smith and Eisner, 2004; Smith and Eisner, 2005) over bootstrapping approaches like self-training (McClosky et al., 2006) to feature-based enhancements of discriminative reranking models (Koo et al., 2008) and the application of semi-supervised SVMs (Wang et al., 2008). The partial correspondence method we present in this paper is compatible with such approaches and can be combined with other weakly supervised machine learning schemes. Our approach is similar to that of Clark and Curran (2006) who use partial training data (CCG lexical categories) for domain adaptation; however, they assume an existing CCG resource for the language in question to provide this data.

3 Projection of Dependency Trees

Most state-of-the-art parsers for natural languages are data-driven and depend on the availability of sufficient amounts of labeled training data. However, manual creation of treebanks is time-consuming and labour-intensive. One way to avoid the expensive annotation process is to automatically label the training data using *annotation projection* (Yarowsky et al., 2001): Given a suitable resource (such as a parser) in language L_1, and a word-aligned parallel corpus with languages L_1 and L_2, label the L_1-portion of the parallel text (with the parser) and copy the annotations to the corresponding (i.e., aligned) elements in language L_2. This is illustrated in Fig. 1a. The arrows between English and Dutch words indicate the word alignment. Assuming we have a parser to produce the dependency tree for the English sentence, we build the tree for the Dutch sentence by establishing arcs between words w_D (e.g., *Ik*) and h_D (*heb*) if there are aligned pairs (w_D, w_E)

	#sents w/ projected parse	avg. sent length	vocab (lemma)
unfiltered	(100,000)	24.92	19,066
bidirectional	2,112	6.39	1,905
fallback	6,426	9.72	4,801
bi+frags$_{\leq 3}$	7,208	9.44	4,631

Table 1: Data reduction effect of noise filters.

(*Ik* and *I*) and (h_D, h_E) (*heb* and *have*) such that h_E is the head of w_E in the English tree.

Annotation projection assumes *direct correspondence* (Hwa et al., 2005) between languages (or annotations), which—although it is valid in many cases—does not hold in general: non-parallelism between corresponding expressions in L_1 and L_2 causes errors in the target annotations. The word alignment constitutes a further source for errors if it is established automatically—which is typically the case in large parallel corpora.

We have implemented a language-independent framework for dependency projection and use the Europarl corpus (Koehn, 2005) as the parallel text. Europarl consists of the proceedings of the European Parliament, professionally translated in 11 languages (approx. 30mln words per language). The data was aligned on the word level with GIZA++ (Och and Ney, 2003).[1] In the experiments reported here, we use the language pair English-Dutch, with English as the source for projection (L_1) and Dutch as L_2. The English portion of the Europarl corpus was lemmatized and POS tagged with the TreeTagger (Schmid, 1994) and then parsed with MaltParser (which is described in Sec. 6), trained on a dependency-converted version of the WSJ part from the Penn Treebank (Marcus et al., 1994), but with the automatic POS tags. The Dutch sentences were only POS tagged (with TreeTagger).[2]

3.1 Data Loss Through Filtering

We quantitatively assess the impact of various filtering techniques on a random sample of 100,000 English-Dutch sentence pairs from Europarl (avg. 24.9 words/sentence). The English dependency trees are projected to their Dutch counterparts as explained above for Fig. 1a.

The first filter we examine is the one that considers exclusively bidirectional alignments. It admits dependency arcs to be projected only if the head h_E and the dependent w_E are each aligned *bidirectionally* with some word in the Dutch sentence. This is indicated in Fig. 1b, where the English verb *are* is aligned with the Dutch translation *heeft* only in one direction. This means that none of the dependencies involving *are* are projected, and the projected structure is not connected. We will discuss in subsequent sections how less restricted projection methods can still incorporate such data.

Table 1 shows the quantitative effect of the bidirectional filter in the row labeled 'bidirectional'. The proportion of usable sentences is reduced to 2.11%. Consequently, the vocabulary size diminishes by a factor of 10, and the average sentence length drops considerably from almost 25 to less than 7 words, suggesting that most non-trivial examples are lost.

3.2 Constrained Fallback Projection

As an instance of a more relaxed projection of complete structures, we also implemented a fallback to unidirectional links which projects further dependencies *after* a partial structure has been built based on the more reliable bidirectional links. That is, the dependencies established via unidirectional alignments are constrained by the existing subtrees, and are subject to the wellformedness conditions for dependency trees.[3] Fig. 1c shows how the fallback mechanism, initialized with the unconnected structure built with the bidirectional filter, recovers a parse tree for the weakly aligned sentence pair in Fig. 1b. Starting with the leftmost word in the Dutch sentence and its English translation (*U* and *You*), there is a unidirectional alignment for the head of *You*: *are* is aligned to *heeft*, so *U* is established as a dependent of *heeft* via fallback. Likewise, *heeft* can now be identified as the root node. Note that the (incorrect) alignment between *heeft* and *You* will not be pursued because it would lead to *heeft* being a dependent of itself and thus violating the wellformed-

[1] Following standard practice, we computed word alignments in both directions ($L_1 \rightarrow L_2$ and $L_2 \rightarrow L_1$); this gives rise to two unidirectional alignments. The *bidirectional alignment* is the intersection of the two unidirectional ones.

[2] The Dutch POS tags are used to train the monolingual parsers from the projected dependency trees (Sec. 7).

[3] I.e., single headedness and acyclicity; we do not require the trees to be projective, but instead train pseudo-projective models (Nivre and Nilsson, 2005) on the projected data (cf. fn. 5).

#frags	1	2	3	4–15	>15
#words					
<4	425	80	12	–	–
4–9	**1,331**	**1,375**	**1,567**	4,793	–
10–19	**339**	**859**	**1,503**	27,910	522
20–30	**17**	**45**	**143**	20,756	10,087
>30	**0**	**5**	**5**	4,813	23,362

Table 2: Fragmented parses projected with the alignment filter. The sentences included in the data set 'bi+frags$_{\leq 3}$' are in boldface.

ness conditions. Finally, the subtree rooted in *gelijk* is incorporated as the second dependent of *heeft*.

As expected, the proportion of examples that pass this filter rises, to 6.42% (Table 1, 'fallback'). However, we will see in Sec. 7 that parsers trained on this data do not improve over parsers trained on the bidirectionally aligned sentences alone. This is presumably due to the noise that inevitably enters the training data through fallback.

4 Partial Correspondence Projection

So far, we have only considered complete trees, i.e., projected structures with exactly one root node. This is a rather strict requirement, given that even state-of-the-art parsers sometimes fail to produce plausible complete analyses for long sentences, and that non-sentential phrases such as complex noun phrases still contain valuable, non-trivial information. We therefore propose *partial correspondence projection* which, in addition to the complete annotations produced by tree-oriented projection, yields partial structures: It admits fragmented analyses in case the tree-oriented projection cannot construct a complete tree. Of course, the nature of those fragments needs to be restricted so as to exclude data with no (interesting) dependencies. E.g., a sentence of five words with a parse consisting of five fragments provides virtually no information about dependency structure. Hence, we impose a limit (fixed at 3 after quick preliminary tests on automatically labeled development data) on the number of fragments that can make up an analysis. Alternatively, one could require a minimum fragment size.

As an example, consider again Fig. 1b. This example would be discarded in strict tree projection, but under partial correspondence it is included as a partial analysis consisting of three fragments:

Although the amount of information provided in this analysis is limited, the arc between *gelijk* and *volkomen*, which is strongly supported by the alignment, can be established without including potentially noisy data points that are only weakly aligned.

We use partial correspondence in combination with bidirectional projection.[4] As can be seen in Table 1 ('bi+frags$_{\leq 3}$'), this combination boosts the amount of usable data to a range similar to that of the fallback technique for trees; but unlike the latter, partial correspondence continues to impose a high-precision filter (bidirectionality) while improving recall through relaxed structural requirements (partial correspondence). Table 2 shows how fragment size varies with sentence length.

5 Data-driven Dependency Parsing

Models for data-driven dependency parsing can be roughly divided into two paradigms: Graph-based and transition-based models (McDonald and Nivre, 2007).

5.1 Graph-based Models

In the graph-based approach, global optimization considers all possible arcs to find the tree $\hat{T}$ s.t.

$$\hat{T} = \arg\max_{T \in D} s(T) = \arg\max_{T \in D} \sum_{(i,j,l) \in A_T} s(i,j,l)$$

where D is the set of all well-formed dependency trees for the sentence, A_T is the set of arcs in T, and $s(i,j,l)$ is the score of an arc between words w_i and w_j with label l. The specific graph-based parser we use in this paper is the MSTParser of McDonald et al. (2005). The MSTParser learns the scoring function s using an online learning algorithm (Crammer and Singer, 2003) which maximizes the margin between $\hat{T}$ and $D \setminus \{\hat{T}\}$, based on a loss function that counts the number of words with incorrect parents relative to the correct tree.

5.2 Transition-based Models

In contrast to the global optimization employed in graph-based models, transition-based models construct a parse tree in a stepwise way: At each point,

[4] Fragments from fallback projection turned out not to be helpful as training data for dependency parsers.

the locally optimal parser action (*transition*) t^* is determined greedily on the basis of the current configuration c (previous actions plus local features):

$$t^* = \arg\max_{t \in T} \; s(c, t)$$

where T is the set of possible transitions. As a representative of the transition-based paradigm, we use the MaltParser (Nivre et al., 2006). It implements incremental, deterministic parsing algorithms and employs SVMs to learn the transition scores s.

6 Parsing with Fragmented Trees

To make effective use of the fragmented trees produced by partial correspondence projection, both parsing approaches need to be adapted for training on sentences with unconnected substructures. Here we briefly discuss how we represent these structures, and then describe how we modified the parsers.

We use the CoNLL-X data format for dependency trees (Buchholz and Marsi, 2006) to encode partial structures. Specifically, every fragment root specifies as its head an artificial root token w_0 (distinguished from a true root dependency by a special relation FRAG). Thus, sentences with a fragmented parse are still represented as a single sentence, including all words; the difference from a fully parsed sentence is that unconnected substructures are attached directly under w_0. For instance, the partial parse in Fig. 1b would be represented as follows (details omitted):

(1)
```
1 U        pron 0 FRAG
2 heeft    verb 0 ROOT
3 volkomen adj  4 mod
4 gelijk   noun 0 FRAG
```

6.1 Graph-based Model: fMST

In the training phase, the MSTParser tries to maximize the scoring margin between the correct parse and all other valid dependency trees for the sentence. However, in the case of fragmented trees, the training example is not strictly speaking correct, in the sense that it does not coincide with the desired parse tree. In fact, this desired tree is among the other possible trees that MST assumes to be incorrect, or at least suboptimal. In order to relax this assumption, we have to ensure that the loss of the desired tree is zero. While it is impossible to single out this one tree (since we do not know which one it is), we can steer the margin in the right direction with a loss function that assigns zero loss to all trees that are *consistent* with the training example, i.e., trees that differ from the training example at most on those words that are fragment roots (e.g., *gelijk* in Fig. 1). To reflect this notion of loss during optimization, we also adjust the definition of the score of a tree:

$$s(T) = \sum_{(i,j,l) \in A_T:\, l \neq \text{FRAG}} s(i, j, l)$$

We refer to this modified model as *f(iltering)MST*.

6.2 Transition-based Model: fMalt

In the transition-based paradigm, it is particularly important to preserve the original context (including unattached words) of a partial analysis, because the parser partly bases its decisions on neighboring words in the sentence.

Emphasis of the role of isolated FRAG dependents as context rather than proper nodes in the tree can be achieved, as with the MSTParser, by eliminating their effect on the margin learned by the SVMs. Since MaltParser scores local decisions, this simply amounts to suppressing the creation of SVM training instances for such nodes (*U* and *gelijk* in (1)). That is, where the feature model refers to context information, unattached words provide this information (e.g., the feature vector for *volkomen* in (1) contains the form and POS of *gelijk*), but there are no instances indicating how they should be attached themselves. This technique of excluding fragment roots during training will be referred to as *fMalt*.

7 Experiments

7.1 Setup

We train instances of the graph- and the transition-based parser on projected dependencies, and occasionally refer to these as "projected parsers".[5]

All results were obtained on the held-out CoNLL-X test set of 386 sentences (avg. 12.9

[5]The MaltParsers use the projective Nivre arc-standard parsing algorithm. For SVM training, data are split on the coarse POS tag, with a threshold of 5,000 instances. MSTParser instances use the projective Eisner parsing algorithm, and first-order features. The input for both systems is projectivized using the head+path schema (Nivre and Nilsson, 2005).

	Malt	MST
Alpino	80.05	82.43
EP	75.33	73.09
Alpino + EP	77.47	81.63
baseline 1 (previous)	23.65	
baseline 2 (next)	27.63	

Table 3: Upper and lower bounds (UAS).

words/sentence) from the Alpino treebank (van der Beek et al., 2002). The Alpino treebank consists mostly of newspaper text, which means that we are evaluating the projected parsers, which are trained on Europarl, in an *out-of-domain* setting, in the absence of manually annotated Europarl test data.

Parsing performance is measured in terms of *unlabeled attachment score (UAS)*, i.e., the proportion of tokens that are assigned the correct head, irrespective of the label.[6]

To establish upper and lower bounds for our task of weakly supervised dependency parsing, we proceed as follows. We train MaltParsers and MSTParsers on (i) the CoNLL-X training portion of the Alpino treebank (195,000 words), (ii) 100,000 Europarl sentences parsed with the parser obtained from (i), and (iii) the concatenation of the data sets (i) and (ii). The first is a supervised upper bound (80.05/82.43% UAS)[7] trained on manually labeled in-domain data, while the second constitutes a weaker bound (75.33/73.09%) subject to the same out-of-domain evaluation as the projected parsers, and the third (77.47%) is a self-trained version of (i). We note in passing that the supervised model does not benefit from self-training. Two simple baselines provide approximations to a lower bound: Baseline 1 attaches every word to the preceding word, achieving 23.65%. Analogously, baseline 2 attaches every word to the following word (27.63%). These systems are summarized in Table 3.

[6]The labeled accuracy of our parsers lags behind the UAS, because the Dutch dependency relations in the projected annotations arise from a coarse heuristic mapping from the original English labels. We therefore report only UAS.

[7]The upper bound models are trained with the same parameter settings as the projected parsers (see fn. 5), which were adjusted for noisy training data. Thus improvements are likely with other settings: Nivre et al. (2006) report 81.35% for a Dutch MaltParser with optimized parameter settings. McDonald et al. (2006) report 83.57% with MST.

		words	Malt	MST
a.	trees (bidirectional)	13,500	65.94	67.76
	trees (fallback)	62,500	59.28	65.08
	bi+frags$_{\leq 3}$	68,000	55.09	57.14
	bi+frags$_{\leq 3}$ (fMalt/fMST)	68,000	**69.15**	**70.02**
b.	trees (bidirectional)	100,000	61.86	**69.91**
	trees (fallback)	100,000	60.05	64.84
	bi+frags$_{\leq 3}$	100,000	54.50	55.87
	bi+frags$_{\leq 3}$ (fMalt/fMST)	100,000	**68.65**	**69.86**
c.	trees (bidirectional)	102,300	63.32	69.85
	trees (fallback)	465,500	53.45	64.88
	bi+frags$_{\leq 3}$	523,000	51.48	57.20
	bi+frags$_{\leq 3}$ (fMalt/fMST)	523,000	**69.52**	**70.33**

Table 4: UAS of parsers trained on projected dependency structures for (a) a sample of 100,000 sentences, subject to filtering, (b) 10 random samples, each with 100,000 words after filtering (average scores given), and (c) the entire Europarl corpus, subject to filtering.

7.2 Results

Table 4a summarizes the results of training parsers on the 100,000-sentence sample analyzed above. Both the graph-based (MST) and the transition-based (Malt) parsers react similarly to the more or less aggressive filtering methods, but to different degrees. The first two rows of the table show the parsers trained on complete trees ('trees (bidirectional)' and 'trees (fallback)'). In spite of the additional training data gained by the fallback method, the resulting parsers do not achieve higher accuracy; on the contrary, there is a drop in UAS, especially in the transition-based model (-6.66%). The increased level of noise in the fallback data has less (but significant)[8] impact on the graph-based counterpart (-2.68%).

Turning to the parsers trained on partial correspondence data ('bi+frags$_{\leq 3}$'), we observe even greater deterioration in both parsing paradigms if the data is used as is. However, in combination with the fMalt/fMST systems ('bi+frags$_{\leq 3}$ (fMalt/fMST)'), both parsers significantly outperform the tree-

[8]Significance testing ($p<.01$) was performed by means of the t-test on the results of 10 training cycles (Table 4c 'trees (fb.)' only 2 cycles due to time constraints). For the experiments in Table 4a and 4c, the cycles differed in terms of the order in which sentences where passed to the parser. In Table 4b we base significance on 10 true random samples for training.

		Recall					Precision				
	dep. length	1	2	3–6	≥7	root	1	2	3–6	≥7	root
a.	trees (bi.)	83.41	66.44	52.94	40.64	52.45	82.46	66.06	61.38	34.95	50.97
	trees (fb.)	82.20	64.21	54.59	37.95	55.72	82.64	61.41	54.39	31.96	68.55
	bi+frags$_{\leq 3}$	70.18	59.50	46.61	32.14	**61.87**	**83.75**	67.22	58.25	32.81	27.01
	bi+frags$_{\leq 3}$ (fMalt)	**89.23**	**75.34**	**59.18**	41.65	59.06	**83.46**	69.05	**65.85**	**48.21**	**75.79**
	Alpino-Malt	92.81	84.94	75.11	65.44	66.15	89.71	81.08	77.56	62.57	84.58
b.	trees (bi.)	**87.53**	**73.79**	59.57	46.79	71.01	86.43	74.08	64.78	45.17	**66.79**
	trees (fb.)	82.53	69.37	55.77	37.46	70.24	85.31	69.29	59.85	40.14	53.99
	bi+frags$_{\leq 3}$	68.11	57.48	34.30	13.00	**90.68**	**90.28**	**78.54**	66.36	43.70	23.41
	bi+frags$_{\leq 3}$ (fMST)	**87.73**	**72.84**	**62.55**	**50.15**	67.78	86.94	71.60	66.05	**48.48**	**68.20**
	Alpino-MST	94.13	86.60	76.91	65.14	71.60	91.76	82.49	76.23	71.96	85.38

Table 5: Performance relative to dependency length. (a) Projected MaltParsers and (b) projected MSTParsers.

oriented models ('trees (bidirectional)') by 3.21% (Malt) and 2.26% (MST).

It would be natural to presume that the superiority of the partial correspondence filter is merely due to the amount of training data, which is larger by a factor of 5.04. We address this issue by isolating the effect on the quality of the data, and hence the success at noise reduction: In Table 4b, we control for the amount of data that is effectively used in training, so that each filtered training set consists of 100,000 words. Considering the Malt models, we find that the trends suggested in Table 4a are confirmed: The pattern of relative performance emerges even though any quantitative (dis-)advantages have been eliminated.[9] [10] Interestingly, the MSTParser does not appear to gain from the increased variety (cf. Table 1) in the partial data: it does not differ significantly from the 'trees (bi.)' model.

Finally, Table 4c provides the results of training on the entire Europarl, or what remains of the corpus after the respective filters have applied. The results corroborate those obtained for the smaller samples.

In summary, the results support our initial hypothesis that partial correspondence for sentences containing a highly reliable part is preferable to relaxing the reliability citerion, and—in the case of the transition-based MaltParser—also to aggressively filtering out all but the reliable complete trees. With UASs around 70%, both systems are only 5% behind the average 75.07% UAS achieved for Dutch in the CoNLL-X Shared Task.

8 Analysis

We have seen that the graph- and the transition-based parser react similarly to the various filtering methods. However, there are interesting differences in the magnitude of the performance changes. If we compare the two tree-oriented filters 'trees (bi.)' and 'trees (fb.)', we observe that, although both Malt and MST suffer from the additional noise that is introduced via the unidirectional alignments, the drop in accuracy is much less pronounced in the latter, graph-based model. Recall that in this paradigm, optimization is performed over the entire tree by scoring edges independenly; this might explain why noisy arcs in the training data have only a negligible impact. Conversely, the transition-based MaltParser, which constructs parse trees in steps of locally optimal decisions, has an advantage when confronted with partial structures: The individual fragments provide exactly the local context, plus lexical information about the (unconnected) wider context.

To give a more detailed picture of the differences between predicted and actual annotations, we show the performance (of the parsers from Table 4b) separately for binned arc length (Table 5) and sentence length (Table 6). As expected, the performance of both the supervised upper bounds (Alpino-

[9] The degree of skewedness in the filtered data is not controlled, as it is an important characteristic of the filters.

[10] Some of the parsers trained on the larger data sets (Table 4b+c) achieve worse results than their smaller counterparts in Table 4a. We conjecture that it is due to the thresholded POS-based data split, performed prior to SVM training: Larger training sets induce decision models with more specialized SVMs, which are more susceptible to tagging errors. This could be avoided by increasing the threshold for splitting.

	sent. length	<4	4–9	10–19	20–30	> 30
a.	trees (bi.)	**73.87**	62.13	65.67	60.81	55.18
	trees (fb.)	69.91	57.84	62.29	60.04	55.47
	bi+frags$_{\leq 3}$	74.14	54.40	56.62	54.07	48.95
	bi+fr$_{\leq 3}$ (fMalt)	**73.51**	**65.69**	**71.70**	**68.49**	**63.71**
	Alpino-Malt	81.98	69.81	81.11	82.82	76.02
b.	trees (bi.)	**76.67**	**70.16**	**73.09**	**69.56**	**63.57**
	trees (fb.)	73.24	64.93	67.79	64.98	57.70
	bi+frags$_{\leq 3}$	77.48	59.65	55.96	55.27	52.74
	bi+fr$_{\leq 3}$ (fMST)	73.24	**67.84**	**73.46**	**70.04**	**62.92**
	Alpino-MST	81.98	72.24	85.10	83.86	78.51

Table 6: UAS relative to sentence length. (a) Projected MaltParsers and (b) projected MSTParsers.

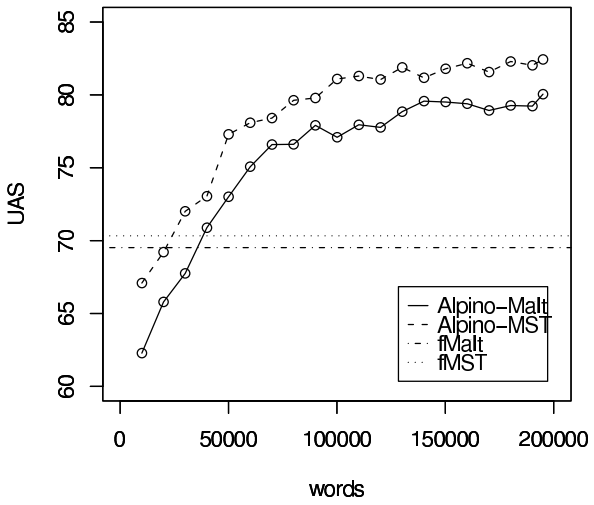


Figure 2: Learning curves for the supervised upper bounds. They reach the performance of the projected parsers with ∼25,000 (MST) resp. 35,000 (Malt) words.

Malt/MST) and the projected parsers degrades as dependencies get longer, and the difference between the two grows. Performance across sentence length remains relatively stable. But note that both tables again reflect the pattern we saw in Table 4. Importantly, the relative ranking (in terms of f-score, not shown, resp. UAS) is still in place even in long distance dependencies and long sentences. This indicates that the effects we have described are not artifacts of a bias towards short dependencies.

In addition, Table 5 sheds some light on the impact of fMalt/fMST in terms of the trade-off between precision and recall. Without the specific adjustments to handle fragments, partial structures in the training data lead to an immense drop in recall. By contrast, when the adapted parsers fMalt/fMST are applied, they boosts recall back to a level comparable to or even above that of the tree-oriented projection parsers, while maintaining precision. Again, this effect can be observed across all arc lengths, except arcs to root, which naturally the 'bi+frags' models are overly eager to predict.

Finally, the learning curves in Fig. 2 illustrate how much labeled data would be required to achieve comparable performance in a supervised setting. The graph-based upper bound (Alpino-MST) reaches the performance of fMST (trained on the entire Europarl) with approx. 25,000 words of manually labeled treebank data; Alpino-Malt achieves the performance of fMalt with approx. 35,000 words. The manual annotation of even these moderate amounts of data involves considerable efforts, including the creation of annotation guidelines and tools, the training of annotators etc.

9 Conclusion

In the context of dependency parsing, we have proposed partial correspondence projection as a greedy method for noise reduction, and illustrated how it can be integrated with data-driven parsing. Our experimental results show that partial tree structures are well suited to train transition-based dependency parsers. Graph-based models do not benefit as much from additional partial structures, but instead are more robust to noisy training data, even when the training set is very small.

In future work, we will explore how well the techniques presented here for English and Dutch work for languages that are typologically further apart, e.g., English-Greek or English-Finnish. Moreover, we are going to investigate how our approach, which essentially ignores unknown parts of the annotation, compares to approaches that marginalize over hidden variables. We will also explore ways of combining graph-based and transition-based parsers along the lines of Nivre and McDonald (2008).

Acknowledgments

The research reported in this paper has been supported by the German Research Foundation DFG as part of SFB 632 "Information structure" (project D4; PI: Kuhn).

References

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of CoNLL-X*, pages 149–164, New York City, June.

Stephen Clark and James R. Curran. 2006. Partial training for a lexicalized-grammar parser. In *Proceedings of HLT-NAACL 2006*, pages 144–151, New York, June.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Reseach*, 3:951–991, January.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(3):311–325.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL 2004*, pages 478–485, Barcelona, Spain.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 127–133.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the MT Summit 2005*.

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

Mitchell Marcus, Grace Kim, Mary Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of HLT-NAACL 2006*, pages 152–159, New York, June.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL 2007*, pages 122–131.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP 2005)*.

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

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

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL 2005*, pages 99–106.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL-X*, pages 221–225.

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

Sebastian Padó and Mirella Lapata. 2006. Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of COLING/ACL 2006*, Sydney, Australia.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of ACL 1992*, pages 128–135.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, pages 44–49, Manchester, England.

Noah A. Smith and Jason Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *Proceedings of ACL 2004*, pages 487–494, Barcelona, July.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of ACL 2005*, pages 354–362, Ann Arbor, MI, June.

Kathrin Spreyer and Anette Frank. 2008. Projection-based acquisition of a temporal labeller. In *Proceedings of IJCNLP 2008*, Hyderabad, India, January.

Leonoor van der Beek, Gosse Bouma, Robert Malouf, and Gertjan van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

Qin Iris Wang, Dale Schuurmans, and Dekang Lin. 2008. Semi-supervised convex training for dependency parsing. In *Proceedings of ACL-HLT 2008*, pages 532–540, Columbus, Ohio, June.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT 2001*.

A metalearning approach to processing the scope of negation

Roser Morante, Walter Daelemans
CNTS - Language Technology Group
University of Antwerp
Prinsstraat 13, B-2000 Antwerpen, Belgium
{Roser.Morante,Walter.Daelemans}@ua.ac.be

Abstract

Finding negation signals and their scope in text is an important subtask in information extraction. In this paper we present a machine learning system that finds the scope of negation in biomedical texts. The system combines several classifiers and works in two phases. To investigate the robustness of the approach, the system is tested on the three subcorpora of the BioScope corpus representing different text types. It achieves the best results to date for this task, with an error reduction of 32.07% compared to current state of the art results.

1 Introduction

In this paper we present a machine learning system that finds the scope of negation in biomedical texts. The system works in two phases: in the first phase, negation signals are identified (i.e., words indicating negation), and in the second phase the full scope of these negation signals is determined. Although the system was developed and tested on biomedical text, the same approach can also be used for text from other domains.

Finding the scope of a negation signal means determining at sentence level the sequence of words in the sentence that is affected by the negation. This task is different from determining whether a word is negated or not. For a sentence like the one in Example (1) taken from the BioScope corpus (Szarvas et al., 2008), the system detects that *lack, neither*, and *nor* are negation signals; that *lack* has as its scope *lack of CD5 expression*, and that the discontinuous negation signal *neither ... nor* has as its scope *neither to segregation of human autosome 11, on which the CD5 gene has been mapped, nor to deletion of the CD5 structural gene*.

(1) <sentence id="S334.5">Analysis at the phenotype and genetic level showed that <xcope id"X334.5.3"><cue type="negation" ref="X334.5.3">lack</cue> of CD5 expression</xcope> was due <xcope id="X334.5.1"> <cue type="negation" ref="X334.5.1">neither</cue> to segregation of human autosome 11, on which the CD5 gene has been mapped, <cue type="negation" ref="X334.5.1">nor</cue> to deletion of the CD5 structural gene</xcope>.</sentence>

Predicting the scope of negation is relevant for text mining and information extraction purposes. As Vincze et al. (2008) put it, extracted information that falls in the scope of negation signals cannot be presented as factual information. It should be discarded or presented separately. Szarvas et al. (2008) report that 13.45% of the sentences in the abstracts section of the BioScope corpus and 12.70% of the sentences in the full papers section contain negations. A system that does not deal with negation would treat the facts in these cases incorrectly as positives. Additionally, information about the scope of negation is useful for entailment recognition purposes.

The approach to the treatment of negation in NLP presented in this paper was introduced in Morante et al. (2008). This system achieved a 50.05 percentage of correct scopes but had a number of important shortcomings. The system presented here uses a different architecture and different classification task definitions, it can deal with multiword negation signals, and it is tested on three subcorpora of the BioScope corpus. It achieves an error reduction of

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 21–29,
Boulder, Colorado, June 2009.

32.07% compared to the previous system.

The paper is organised as follows. In Section 2, we summarise related work. In Section 3, we describe the corpus on which the system has been developed. In Section 4, we introduce the task to be performed by the system, which is described in Section 5. Results are presented and discussed in Section 6. Finally, Section 7 puts forward some conclusions.

2 Related work

Negation has been a neglected area in open-domain natural language processing. Most research has been performed in the biomedical domain and has focused on detecting whether a medical term is negated or not, whereas in our approach we focus on detecting the full scope of negation signals.

Chapman et al. (2001) developed NegEx, a regular expression based algorithm for determining whether a finding or disease mentioned within narrative medical reports is present or absent. The reported results are 94.51% precision and 77.84% recall. Mutalik et al. (2001) developed Negfinder, a rule-based system that recognises negated patterns in medical documents. It consists of two tools: a lexical scanner that uses regular expressions to generate a finite state machine, and a parser. The reported results are 95.70% recall and 91.80% precision.

Sanchez-Graillet and Poesio (2007) present an analysis of negated interactions in 50 biomedical articles and a heuristics-based system that extracts such information. The preliminary results reported range from 54.32% F-score to 76.68%, depending on the method applied. Elkin et al. (2005) describe a rule-based system that assigns to concepts a level of certainty as part of the generation of a dyadic parse tree in two phases: First a preprocessor breaks each sentence into text and operators. Then, a rule based system is used to decide if a concept has been positively, negatively, or uncertainly asserted. The system achieves 97.20% recall and 98.80% precision.

The systems mentioned above are essentially based on lexical information. Huang and Lowe (2007) propose a classification scheme of negations based on syntactic categories and patterns in order to locate negated concepts, regardless of their distance from the negation signal. Their hybrid system that combines regular expression matching with grammatical parsing achieves 92.60% recall and 99.80% precision. Additionally, Boytcheva et al. (2005) incorporate the treatment of negation in a system, MEHR, that extracts from electronic health records all the information required to generate automatically patient chronicles. They report 57% of negations correctly recognised.

The above-mentioned research applies rule-based algorithms to negation finding. Machine learning techniques have been used in some cases. Averbuch et al. (2004) developed an algorithm that uses information gain to learn negative context patterns. Golding and Chapman (2003) experiment with Naive Bayes and Decision Trees to distinguish whether a medical observation is negated by the word *not* in a corpus of hospital reports. They report a maximum of 90% F-score.

Goryachev et al. (2006) compare the performance of four different methods of negation detection, two regular expression-based methods and two classification-based methods trained on 1745 discharge reports. They show that the regular expression-based methods show better agreement with humans and better accuracy than the classification methods. Like in most of the work mentioned, the task consists in determining whether a medical term is negated. Rokach et al. (2008) present a new pattern-based algorithm for identifying context in free-text medical narratives.The originality of the algorithm lies in that it automatically learns patterns similar to the manually written patterns for negation detection.

We are not aware of any research that has focused on learning the full scope of negation signals outside biomedical natural language processing.

3 Negation in the BioScope Corpus

The system has been developed using the BioScope corpus (Szarvas et al., 2008; Vincze et al., 2008)[1], a freely available resource that consists of medical and biological texts. In the corpus, every sentence is annotated with information about negation and speculation. The annotation indicates the boundaries of the scope and the keywords, as shown in (1) above. In the annotation, scopes are extended to the

[1]Web page: `www.inf.u-szeged.hu/rgai/bioscope`.

biggest syntactic unit possible, so that scopes have the maximal length, and the negation signal is always included in the scope. The annotation guidelines and the inter-annotator agreement information can be found on the web page.

	Clinical	Papers	Abstracts
#Documents	1954	9	1273
#Sentences	6383	2670	11871
#Words	41985	60935	282243
#Lemmas	2320	5566	14506
Av. length sentences	7.73	26.24	26.43
% Sent. 1-10 tokens	75.85	11.27	3.17
% Sent. 11-20 tokens	20.99	27.67	30.49
% Sent. 21-30 tokens	2.94	29.55	35.93
% Sent. 31-40 tokens	0.15	17.00	19.76
% Sent. $>$ 40 tokens	0.01	0.03	10.63
%Negation sentences	13.55	12.70	13.45
#Negation signals	877	389	1848
Av. length scopes	4.98	8.81	9.43
Av. length scopes to the right	4.84	7.61	8.06
Av. length scopes to the left	6.33	5.69	8.55
% Scopes to the right	97.64	81.77	85.70
% Scopes to the left	2.35	18.22	14.29

Table 1: Statistics about the subcorpora in the BioScope corpus and the negation scopes ("Av". stands for *average*).

The BioScope corpus consists of three parts: clinical free-texts (radiology reports), biological full papers and biological paper abstracts from the GENIA corpus (Collier et al., 1999). Table 1 shows statistics about the corpora. Negation signals are represented by one or more tokens.

Only one negation signal (*exclude*) that occurs in the papers subcorpus does not occur in the abstracts subcorpus, and six negation signals (*absence of, exclude, favor, favor over, may, rule out* that appear in the clinical subcorpus do not appear in the abstracts subcorpus. The negation signal *no* (determiner) accounts for 11.74 % of the negation signals in the abstracts subcorpus, 12.88 % in the papers subcorpus, and 76.65 % in the clinical subcorpus. The negation signal *not* (adverb) accounts for 58.89 % of the negation signals in the abstracts subcorpus, 53.22 % in the papers subcorpus, and 6.72 % in the clinical subcorpus.

The texts have been processed with the GENIA tagger (Tsuruoka and Tsujii, 2005; Tsuruoka et al., 2005), a bidirectional inference based tagger that analyzes English sentences and outputs the base forms, part-of-speech tags, chunk tags, and named entity tags in a tab-separated format. Additionally, we converted the annotation about scope of negation into a token-per-token representation, following the standard format of the 2006 CoNLL Shared Task (Buchholz and Marsi, 2006), where sentences are separated by a blank line and fields are separated by a single tab character. A sentence consists of a sequence of tokens, each one starting on a new line.

4 Finding the scope of negation

We model the scope finding task as two consecutive classification tasks: a first one that consists of classifying the tokens of a sentence as being at the beginning of a negation signal, inside or outside. This allows the system to find multiword negation signals.

The second classification task consists of classifying the tokens of a sentence as being the first element of the scope, the last, or neither. This happens as many times as there are negation signals in the sentence. We have chosen this classification model after experimenting with two additional models that produced worse results: in one case we classifed tokens as being inside or outside of the scope. In another case we classified chunks, instead of tokens, as being inside or outside of the scope.

5 System description

The two classification tasks (identifying negation signals and finding the scope) are implemented using supervised machine learning methods trained on part of the annotated corpus.

5.1 Identifying negation signals

In this phase, a classifier predicts whether a token is the first token of a negation signal, inside a negation signal, or outside of it. We use IGTREE as implemented in TiMBL (version 6.1.2) (Daelemans et al., 2007). TiMBL[2] is a software package that contains implementations of memory-based learning algorithms like IB1 and IGTREE. We also experimented with IB1, but it produced lower results.

[2]TiMBL can be downloaded from the web page `http://ilk.uvt.nl/timbl/`.

The classifier was parameterised by using gain ratio for feature weighting. The instances represent all tokens in the corpus and they have features of the token (lemma) and of the token context: word form, POS, and chunk IOB tag[3] of one token to the left and to the right; word form of the second token to the left and to the right. According to the gain ratio scores, the most informative feature is the lemma of the token, followed by the chunk IOB tag of the token to the right, and the features relative to the token to the left.

The test file is preprocessed using a list of negation signals extracted from the training corpus, that are unambiguous in the training corpus. The list comprises the following negation signals: *absence, absent, fail, failure, impossible, lack, loss, miss, negative, neither, never, no, none, nor, not, unable, without*. Instances with this negation signals are directly assigned their class. The classifier predicts the class of the rest of tokens.

5.2 Scope finding

In this phase three classifiers predict whether a token is the first token in the scope sequence, the last, or neither. A fourth classifier is a metalearner that uses the predictions of the three classifiers to predict the scope classes. The three object classifiers that provide input to the metalearner were trained using the following machine learning methods:

- Memory-based learning as implemented in TiMBL (version 6.1.2) (Daelemans et al., 2007), a supervised inductive algorithm for learning classification tasks based on the k-nearest neighbor classification rule (Cover and Hart, 1967). In this lazy learning approach, all training data is kept in memory and classification of a new item is achieved by extrapolation from the most similar remembered training items.
- Support vector machines (SVM) as implemented in SVM^{light}V6.01 (Joachims, 1999). SVMs are defined on a vector space and try to find a decision surface that best separates the data points into two classes. This is achieved by using quadratic programming techniques. Kernel functions can be used to map the original vectors to a higher-dimensional space that is linearly separable.
- Conditional random fields (CRFs) as implemented in CRF++-0.51 (Lafferty et al., 2001). CRFs define a conditional probability distribution over label sequences given a particular observation sequence rather than a joint distribution over label and observation sequences, and are reported to avoid the label bias problem of HMMs and other learning approaches.

The memory-based learning algorithm was parameterised by using overlap as the similarity metric, gain ratio for feature weighting, using 7 k-nearest neighbors, and weighting the class vote of neighbors as a function of their inverse linear distance. The SVM was parameterised in the learning phase for classification, cost factor of 1 and biased hyperplane, and it used a linear kernel function. The CRFs classifier used regularization algorithm L2 for training, the hyper-parameter and the cut-off threshold of features were set to 1.

An instance represents a pair of a negation signal and a token from the sentence. This means that all tokens in a sentence are paired with all negation signals that occur in the sentence. Negation signals are those that have been classified as such in the previous phase. Only sentences that have negation signals are selected for this phase.

We started with a larger, extensive pool of 131 features which encoded information about the negation signal, the paired token, their contexts, and the tokens in between. Feature selection experiments were carried out with the memory-based learning classifier. Features were selected based on their gain ratio, starting with all the features and eliminating the least informative features. We also performed experiments applying the feature selection process reported in Tjong Kim Sang et al. (2005), a bi-directional hill climbing process. However, experiments with this method did not produce a better selection of features.

The features of the first three classifers are:

- Of the negation signal: Chain of words.
- Of the paired token: Lemma, POS, chunk IOB tag, type of chunk; lemma of the second and third tokens to the left; lemma, POS, chunk IOB tag, and type of chunk of the first token to the left and three tokens to the right; first word, last word, chain of words, and chain of POSs of the chunk of the paired token and of two chunks to the left and two chunks to the

[3] Tags produced by the GENIA tagger that indicate if a token is inside a certain chunk, outside, or at the beginning.

right.

- Of the tokens between the negation signal and the token in focus: Chain of POS types, distance in number of tokens, and chain of chunk IOB tags.
- Others: A feature indicating the location of the token relative to the negation signal (pre, post, same).

The fourth classifier, a metalearner, is also a CRF as implemented in CRF++. The features of this classifier are:

- Of the negation signal: Chain of words, chain of POS, word of the two tokens to the right and two tokens to the left, token number divided by the total number of tokens in the sentence.
- Of the paired token: Lemma, POS, word of two tokens to the right and two tokens to the left, token number divided by the total number of tokens in the sentence.
- Of the tokens between the negation signal and the token in focus: Binary features indicating if there are commas, colons, semicolons, verbal phrases or one of the following words between the negation signal and the token in focus:

  Whereas, but, although, nevertheless, notwithstanding, however, consequently, hence, therefore, thus, instead, otherwise, alternatively, furthermore, moreover.
- About the predictions of the three classifiers: prediction, previous and next predictions of each of the classifiers, full sequence of previous and full sequence of next predictions of each of the classifiers.
- Others: A feature indicating the location of the token relative to the negation signal (pre, post, same).

Negation signals in the BioScope corpus always have one consecutive block of scope tokens, including the signal token itself. However, the classifiers only predict the first and last element of the scope. We need to process the output of the classifers in order to build the complete sequence of tokens that constitute the scope. We apply the following post-processing:

(2) - If one token has been predicted as FIRST and one as LAST, the sequence is formed by the tokens between first and last.

- If one token has been predicted as FIRST and none has been predicted as LAST, the sequence is formed by the token predicted as FIRST.

- If one token has been predicted as LAST and none as FIRST, the sequence will start at the negation signal and it will finish at the token predicted as LAST.

- If one token has been predicted as FIRST and more than one as LAST, the sequence will end with the first token predicted as LAST after the token predicted as FIRST, if there is one.

- If one token has been predicted as LAST and more than one as FIRST, the sequence will start at the negation signal.

- If no token has been predicted as FIRST and more than one as LAST, the sequence will start at the negation signal and will end at the first token predicted as LAST after the negation signal.

6 Results

The results provided for the abstracts part of the corpus have been obtained by performing 10-fold cross validation experiments, whereas the results provided for papers and clinical reports have been obtained by training on the full abstracts subcorpus and testing on the papers and clinical reports subcorpus. The latter experiment is therefore a test of the robustness of the system when applied to different text types within the same domain.

The evaluation is made using the precision and recall measures (Van Rijsbergen, 1979), and their harmonic mean, F-score. In the negation finding task, a negation token is correctly classified if it has been classified as being at the beginning or inside the negation signal. We also evaluate the percentage of negation signals that have been correctly identified. In the scope finding task, a token is correctly classified if it has been correctly classified as being inside or outside of the scope of all the negation signals that there are in the sentence. This means that when there is more than one negation signal in the sentence, the token has to be correctly assigned a class for as many negation signals as there are. Additionally, we evaluate the percentage of correct scopes (PCS). A scope is correct if all the tokens in the sentence have been assigned the correct scope class for a specific negation signal. The evaluation in terms of precision and recall measures takes as unit a token, whereas the evaluation in terms of PCS takes as unit a scope.

6.1 Negation signal finding

An informed baseline system has been created by tagging as negation signals the tokens with the words: *absence, absent, fail, failure, impossible, instead of, lack, loss, miss, negative, neither, never, no, none, nor, not, rather than, unable, with the exception of, without*. The list has been extracted from the training corpus. Baseline results and inter-annotator agreement scores are shown in Table 2.

Corpus	Prec.	Recall	F1	Correct	IAA
Abstracts	100.00	95.17	97.52	95.09	91.46
Papers	100.00	92.46	96.08	92.15	79.42
Clinical	100.00	97.53	98.75	97.72	90.70

Table 2: Baseline results of the negation finding system and inter-annotator agreement (IAA) in %.

Table 3 shows the results of the system, which are significantly higher than the results of the baseline system. With a more comprehensive list of negation signals it would be possible to identify all of them in a text.

Corpus	Prec.	Recall	F1	Correct
Abstracts	100.00	98.75	99.37	98.68
Papers	100.00	95.72	97.81	95.80
Clinical	100.00	98.09	99.03	98.29

Table 3: Results of the negation finding system in %.

The lower result of the papers subcorpus is caused by the high frequency of the negation signal *not* in this corpus (53.22 %), that is correct in 93.68 % of the cases. The same negation signal is also frequent in the abstracts subcorpus (58.89 %), but in this case it is correct in 98.25 % of the cases. In the clinical subcorpus *not* has low frequency (6.72 %), which means that the performance of the classifer for this negation signal (91.22 % correct) does not affect so much the global results of the classifier. Most errors in the classification of *not* are caused by the system predicting it as a negation signal in cases not marked as such in the corpus. The following sentences are some examples:

(3) However, programs for tRNA identification [...] do not necessarily perform well on unknown ones.
The evaluation of this ratio is difficult because not all true interactions are known.
However, the Disorder module does not contribute significantly to the prediction.

6.2 Scope finding

An informed baseline system has been created by calculating the average length of the scope to the right of the negation signal in each corpus and tagging that number of tokens as scope tokens. We take the scope to the right for the baseline because it is much more frequent than the scope to the left, as is shown by the statistics contained in Table 1 of Section 3.

Corpus	Prec.	Recall	F1	PCS	PCS-2	IAA
Abstracts	76.68	78.26	77.46	7.11	37.45	92.46
Papers	69.34	66.92	68.11	4.76	24.86	70.86
Clinical	86.85	74.96	80.47	12.95	62.27	76.29

Table 4: Baseline results of the scope finding system and inter-annotator agreement (IAA) in %.

Baseline results and inter-annotator agreement scores are presented in Table 4. The percentage of correct scopes has been measured in two ways: PCS measures the proportion of correctly classified tokens in the scope sequence, whereas PCS-2 measures the proportion of nouns and verbs that are correctly classifed in the scope sequence. This less strict way of computing correctness is motivated by the fact that being able to determine the concepts and relations that are negated (indicated by content words) is the most important use of the negation scope finder. The low PCS for the three subcorpora indicates that finding the scope of negations is not a trivial task. The higher PCS for the clinical subcorpus follows a trend that applies also to the results of the system. The fact that, despite a very low PCS, precision, recall and F1 are relatively high indicates that these measures are in themselves not reliable to evaluate the performance of the system.

The upper-bound results of the metalearner system assuming gold standard identification of negation signals are shown in Table 5.

Corpus	Prec.	Recall	F1	PCS	PCS-2
Abstracts	90.68	90.68	90.67	73,36	74.10
Papers	84.47	84.95	84.71	50.26	54.23
Clinical	91.65	92.50	92.07	87.27	87.95

Table 5: Results of the scope finding system with gold-standard negation signals.

The results of the metalearner system are presented in Table 6. Results with gold-standard nega-

tion signals are especially better for the clinical subcorpus because except for *lack, negative* and *not*, all negation signals score a PCS higher than 90 %. Thus, in the clinical subcorpus, if the negation signals are identified, their scope will be correctly found. This does not apply to the abstracts and papers subcorpus.

Corpus	Prec.	Recall	F1	PCS	PCS-2
Abstracts	81.76	83.45	82.60	66.07	66.93
Papers	72.21	69.72	70.94	41.00	44.44
Clinical	86.38	82.14	84.20	70.75	71.21

Table 6: Results of the scope finding system with predicted negation signals.

In terms of PCS, results are considerably higher than baseline results, whereas in terms of precision, recall and F1, results are slightly higher. Compared to state of the art results (50.05 % PCS in (anonymous reference) for the abstracts subcorpus), the system achieves an error reduction of 32.07 %, which shows that the system architecture presented in this paper leads to more accurate results.

Evaluating the system in terms of a more relaxed measure (PCS-2) does not reflect a significant increase in its performance. This suggests that when a scope is incorrectly predicted, main content tokens are also incorrectly left out of the scope or added. An alternative to the PCS-2 measure would be to mark in the corpus the relevant negated content words and evaluate if they are under the scope.

Results also show that the system is portable to different types of documents, although performance varies depending on the characteristics of the corpus. Clinical reports are easier to process than papers and abstracts, which can be explained by several factors. One factor is the length of sentences: 75.85 % of the sentences in the clinical reports have 10 or less words, whereas this rate is 3.17 % for abstracts and 11.27 % for papers. The average length of a sentence for clinical reports is 7.73 tokens, whereas for abstracts it is 26.43 and for papers 26.24. Shorter sentences imply shorter scopes. In the scope finding phase, when we process the output of the classifiers to build the complete sequence of tokens that constitute the scope, we give preference to short scopes by choosing as LAST the token classified as LAST that is the closest to the negation signal. A way to make the system better portable to texts with longer sentences would be to optimise the choice of the last token in the scope.

	Abstracts		Papers		Clinical	
	#	PCS	#	PCS	#	PCS
absence	57	56.14	-	-	-	-
absent	13	15.38	-	-	-	-
can not	28	42.85	16	50.00	-	-
could not	14	57.14	-	-	-	-
fail	57	63.15	13	38.46	-	-
lack	85	57.64	20	45.00	-	-
negative	-	-	-	-	17	0.00
neither	33	51.51	-	-	-	-
no	**207**	**73.42**	**44**	**50.00**	**673**	**73.10**
nor	43	44.18	-	-	-	-
none	7	57.14	10	0.00	-	-
not	**1036**	**69.40**	**200**	**39.50**	**57**	**50.87**
rather than	20	65.00	12	41.66	-	-
unable	30	40.00	-	-	-	-
without	82	89.02	24	58.33	-	-

Table 7: PCS per negation signal for negation signals that occur more than 10 times in one of the subcorpus.

Another factor that causes a higher performance on the clinical subcorpus is the frequency of the negation signal *no* (76.65 %), which has also a high PCS in abstracts, as shown in Table 7. Typical example sentences with this negation signal are shown in (4). Its main characteristics are that the scope is very short (5 tokens average in clinical reports) and that it scopes to the right over a noun phrase.

(4) No findings to account for symptoms.
No signs of tuberculosis.

The lower performance of the system on the papers subcorpus compared to the abstracts subcorpus is due to the high proportion of the negation signal *not* (53.22 %), which scores a low PCS (39.50), as shown in Table 7. Table 7 also shows that, except for *can not*, all negation signals score a lower PCS on the papers subcorpus. This difference can not be caused by the sentence length, since the average sentence length in the abstracts subcorpus (26.43 tokens) is similar to the average sentence length in the papers subcorpus (26.24). The difference may be related to the difference in the length of the scopes and their direction. For example, the average length of the scope of *not* is 8.85 in the abstracts subcorpus and 6.45 in the papers subcorpus. The scopes to the

left for *not* amount to 23.28 % in the papers subcorpus and to 16.41 % in the abstracts subcorpus, and the average scope to the left is 5.6 tokens in the papers subcorpus and 8.82 in the abstracts subcorpus.

As for the results per negation signal on the abstracts corpus, the negation signals that score higher PCS have a low (*none*) or null (*absence, fail, lack, neither, no, rather than, without*) percentage of scopes to the left. An exception is *not* with a high score and 16.41% of scopes to the left. The negation signals with lower PCS have a higher percentage of scopes to the left (*absent, can not, nor, unable*). A typical error for the negation signal *unable* is exemplified by the sentence *VDR DNA-binding mutants were unable to either bind to this element in vitro or repress in vivo*, in which the gold scope starts at the beginning of the sentence, where the predicted scopes starts at the negation signal.

6.2.1 Results of the metalearner versus results of the first three classifiers

The choice of a metalearner approach has been motivated by the significantly higher results that the metalearner produces compared to the results of the first three classifiers. The results of each of the classifiers independently are presented in Table 8.

Algor.	Ev.	Abstracts	Papers	Clinical
TiMBL	Prec.	78.85	68.66	82.25
	Rec.	80.54	66.29	78.56
	F1	79.69	67.46	80.36
	PCS	56.80	33.59	**70.87**
	PCS-2	57.99	37.30	71.21
CRF	Prec.	78.49	68.94	93.42
	Rec.	80.16	66.57	80.24
	F1	79.31	67.73	86.33
	PCS	59.90	36.50	59.51
	PCS-2	60.04	38.88	59.74
SVM	Prec.	77.74	68.01	93.80
	Rec.	79.35	65.66	85.16
	F1	78.54	66.82	89.27
	PCS	56.80	33.33	**82.45**
	PCS-2	57.59	35.18	82.68

Table 8: Results for the first three classifiers of the scope finding system.

PCS results show that the metalearner system performs significantly better than the three classifiers for the abstracts and papers subcorpora, but not for the clinical subcorpus, in which case TiMBL and SVM produce higher scores, although only the SVM results are significantly better with a difference of 11.7 PCS. An analysis in detail of the SVM scores per negation signal shows that the main difference between the scores of the metalearner and SVM is that the SVM is good at predicting the scopes of the negation signal *no* when it occurs as the first token in the sentence, like in (4) above. When *no* occurs in other positions, SVM scores 1.17 PCS better.

We plan to perform experiments with the three classifiers using the features of the metalearner that are not related to the predictions, in order to check if the three classifiers would perform better.

7 Conclusions

In this paper we have presented a metalearning approach to processing the scope of negation signals. Its performance is evaluated in terms of percentage of correct scopes on three test sets. With 66.07 % PCS on the abstracts corpus the system achieves 32.07 % of error reduction over current state of the art results. The architecture of the system is new for this problem, with three classifiers and a metalearner that takes as input the output of the first classifiers. The classification task definition is also original.

We have shown that the system is portable to different corpora, although performance fluctuates depending on the characteristics of the corpora. The results per corpus are determined to a certain extent by the scores of the negation signals *no* and *not*, that are very frequent and difficult to process in some text types. Shorter scopes are easier to learn as reflected in the results of the clinical corpus, where *no* is the most frequent negation signal. We have also shown that the metalearner performs better than the three first classifiers, except for the negation signal *no* in clinical reports, for which the SVM classifier produces the highest scores.

Future research will deal with a more detailed analysis of the errors by each of the three initial classifiers compared to the errors of the metalearner in order to better understand why the results of the metalearner are higher. We also would like to perform feature analysis, and test the system on general domain corpora.

Acknowledgments

Our work was made possible through financial support from the University of Antwerp (GOA project BIOGRAPH). We are grateful to four anonymous reviewers for their valuable comments and suggestions.

References

M. Averbuch, T. Karson, B. Ben-Ami, O. Maimon, and L. Rokach. 2004. Context-sensitive medical information retrieval. In *Proc. of the 11th World Congress on Medical Informatics (MEDINFO-2004)*, pages 1–8, San Francisco, CA. IOS Press.

S. Boytcheva, A. Strupchanska, E. Paskaleva, and D. Tcharaktchiev. 2005. Some aspects of negation processing in electronic health records. In *Proc. of International Workshop Language and Speech Infrastructure for Information Access in the Balkan Countries*, pages 1–8, Borovets, Bulgaria.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of the X CoNLL Shared Task*, New York. SIGNLL.

W. W. Chapman, W. Bridewell, P. Hanbury, G. F. Cooper, and B.G. Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *J Biomed Inform*, 34:301–310.

N. Collier, H.S. Park, N. Ogata, Y. Tateisi, C. Nobata, T. Sekimizu, H. Imai, and J. Tsujii. 1999. The GENIA project: corpus-based knowledge acquisition and information extraction from genome research papers. In *Proceedings of EACL-99*.

T. M. Cover and P. E. Hart. 1967. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27.

W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2007. TiMBL: Tilburg memory based learner, version 6.1, reference guide. Technical Report Series 07-07, ILK, Tilburg, The Netherlands.

P. L. Elkin, S. H. Brown, B. A. Bauer, C.S. Husser, W. Carruth, L.R. Bergstrom, and D. L. Wahner-Roedler. 2005. A controlled trial of automated classification of negation from clinical notes. *BMC Medical Informatics and Decision Making*, 5(13).

l. M. Goldin and W.W. Chapman. 2003. Learning to detect negation with 'Not' in medical texts. In *Proceedings of ACM-SIGIR 2003*.

S. Goryachev, M. Sordo, Q.T. Zeng, and L. Ngo. 2006. Implementation and evaluation of four different methods of negation detection. Technical report, DSG.

Y. Huang and H.J. Lowe. 2007. A novel hybrid approach to automated negation detection in clinical radiology reports. *J Am Med Inform Assoc*, 14(3):304–311.

T. Joachims, 1999. *Advances in Kernel Methods - Support Vector Learning*, chapter Making large-Scale SVM Learning Practical, pages 169–184. MIT-Press, Cambridge, MA.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML 2001*, pages 282–289.

R. Morante, A. Liekens, and W. Daelemans. 2008. A combined memory-based semantic role labeler of english. In *Proc. of the EMNLP 2008*, pages 715–724, Honolulu, Hawaii.

A.G. Mutalik, A. Deshpande, and P.M. Nadkarni. 2001. Use of general-purpose negation detection to augment concept indexing of medical documents. a quantitative study using the UMLS. *J Am Med Inform Assoc*, 8(6):598–609.

L. Rokach, R.Romano, and O. Maimon. 2008. Negation recognition in medical narrative reports. *Information Retrieval Online.*

O. Sanchez-Graillet and M. Poesio. 2007. Negation of protein-protein interactions: analysis and extraction. *Bioinformatics*, 23(13):424–432.

G. Szarvas, V. Vincze, R. Farkas, and J. Csirik. 2008. The BioScope corpus: annotation for negation, uncertainty and their scopein biomedical texts. In *Proc. of BioNLP 2008*, pages 38–45, Columbus, Ohio, USA. ACL.

E. Tjong Kim Sang, S. Canisius, A.van den Bosch, and T. Bogers. 2005. Applying spelling error correction techniques for improving semantic role labelling. In *Proc. of CoNLL 2005*, pages 229–232.

Y. Tsuruoka and J. Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proc. of HLT/EMNLP 2005*, pages 467–474.

Y. Tsuruoka, Y. Tateishi, J. Kim, T. Ohta, J. McNaught, S. Ananiadou, and J. Tsujii, 2005. *Advances in Informatics - 10th Panhellenic Conference on Informatics*, volume 3746 of *Lecture Notes in Computer Science*, chapter Part-of-Speech Tagger for Biomedical Text, Advances in Informatics, pages 382–392. Springer, Berlin/Heidelberg.

C.J. Van Rijsbergen. 1979. *Information Retrieval*. Butterworths, London.

V. Vincze, G. Szarvas, R. Farkas, G. Móra, and J. Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9((Suppl 11)):S9.

Efficient Linearization of Tree Kernel Functions

Daniele Pighin
FBK-Irst, HLT
Via di Sommarive, 18 I-38100 Povo (TN) Italy
pighin@fbk.eu

Alessandro Moschitti
University of Trento, DISI
Via di Sommarive, 14 I-38100 Povo (TN) Italy
moschitti@disi.unitn.it

Abstract

The combination of Support Vector Machines with very high dimensional kernels, such as string or tree kernels, suffers from two major drawbacks: first, the implicit representation of feature spaces does not allow us to understand which features actually triggered the generalization; second, the resulting computational burden may in some cases render unfeasible to use large data sets for training. We propose an approach based on feature space reverse engineering to tackle both problems. Our experiments with Tree Kernels on a Semantic Role Labeling data set show that the proposed approach can drastically reduce the computational footprint while yielding almost unaffected accuracy.

1 Introduction

The use of Support Vector Machines (SVMs) in supervised learning frameworks is spreading across different communities, including Computational Linguistics and Natural Language Processing, thanks to their solid mathematical foundations, efficiency and accuracy. Another important reason for their success is the possibility of using kernel functions to implicitly represent examples in some high dimensional kernel space, where their similarity is evaluated. Kernel functions can generate a very large number of features, which are then weighted by the SVM optimization algorithm obtaining a feature selection side-effect. Indeed, the weights encoded by the gradient of the separating hyperplane learnt by the SVM implicitly establish a ranking between features in the kernel space. This property has been exploited in feature selection models based on approximations or transformations of the gradient, e.g. (Rakotomamonjy, 2003), (Weston et al., 2003) or (Kudo and Matsumoto, 2003).

However, kernel based systems have two major drawbacks: first, new features may be discovered in the implicit space but they cannot be directly observed. Second, since learning is carried out in the dual space, it is not possible to use the faster SVM or perceptron algorithms optimized for linear spaces. Consequently, the processing of large data sets can be computationally very expensive, limiting the use of large amounts of data for our research or applications.

We propose an approach that tries to fill in the gap between explicit and implicit feature representations by 1) selecting the most relevant features in accordance with the weights estimated by the SVM and 2) using these features to build an explicit representation of the kernel space. The most innovative aspect of our work is the attempt to model and implement a solution in the context of structural kernels. In particular we focus on Tree Kernel (TK) functions, which are especially interesting for the Computational Linguistics community as they can effectively encode rich syntactic data into a kernel-based learning algorithm. The high dimensionality of a TK feature space poses interesting challenges in terms of computational complexity that we need to address in order to come up with a viable solution. We will present a number of experiments carried out in the context of Semantic Role Labeling, showing that our approach can noticeably reduce training time while yielding almost unaffected classification accuracy, thus allowing us to handle larger data sets at a reasonable computational cost.

The rest of the paper is structured as follows: Sec-

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 30–38,
Boulder, Colorado, June 2009.

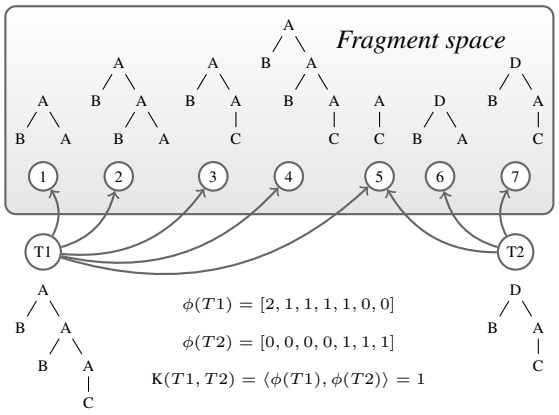


Figure 1: Esemplification of a fragment space and the kernel product between two trees.

tion 2 will briefly review SVMs and Tree Kernel functions; Section 3 will detail our proposal for the linearization of a TK feature space; Section 4 will review previous work on related subjects; Section 5 will describe our experiments and comment on their results; finally, in Section 6 we will draw our conclusions.

2 Tree Kernel Functions

The decision function of an SVM is:

$$f(\vec{x}) = \vec{w} \cdot \vec{x} + b = \sum_{i=1}^{n} \alpha_i y_i \vec{x_i} \cdot \vec{x} + b \qquad (1)$$

where $\vec{x}$ is a classifying example and $\vec{w}$ and b are the separating hyperplane's *gradient* and its *bias*, respectively. The gradient is a linear combination of the training points $\vec{x_i}$, their labels y_i and their weights α_i. These and the bias are optimized at training time by the learning algorithm. Applying the so-called *kernel trick* it is possible to replace the scalar product with a *kernel function* defined over pairs of *objects*:

$$f(o) = \sum_{i=1}^{n} \alpha_i y_i \mathrm{k}(o_i, o) + b$$

with the advantage that we do not need to provide an explicit mapping $\phi(\cdot)$ of our examples in a vector space.

A Tree Kernel function is a convolution kernel (Haussler, 1999) defined over pairs of trees. Practically speaking, the kernel between two trees evaluates the number of substructures (or *fragments*) they have in common, i.e. it is a measure of their overlap. The function can be computed recursively in closed form, and quite efficient implementations are available (Moschitti, 2006). Different TK functions are characterized by alternative fragment definitions, e.g. (Collins and Duffy, 2002) and (Kashima and Koyanagi, 2002). In the context of this paper we will be focusing on the SubSet Tree (SST) kernel described in (Collins and Duffy, 2002), which relies on a fragment definition that does not allow to break production rules (i.e. if any child of a node is included in a fragment, then also all the other children have to). As such, it is especially indicated for tasks involving constituency parsed texts.

Implicitly, a TK function establishes a correspondence between distinct fragments and dimensions in some *fragment space*, i.e. the space of all the possible fragments. To simplify, a tree t can be represented as a vector whose attributes count the occurrences of each fragment within the tree. The kernel between two trees is then equivalent to the scalar product between pairs of such vectors, as exemplified in Figure 1.

3 Mining the Fragment Space

If we were able to efficiently mine and store in a dictionary all the fragments encoded in a model, we would be able to represent our objects explicitly and use these representations to train larger models and very quick and accurate classifiers. What we need to devise are strategies to make this approach convenient in terms of computational requirements, while yielding an accuracy comparable with direct tree kernel usage.

Our framework defines five distinct activities, which are detailed in the following paragraphs.

Fragment Space Learning (*FSL*) First of all, we can partition our training data into S smaller sets, and use the SVM and the SST kernel to learn S models. We will use the estimated weights to drive our feature selection process. Since the time complexity of SVM training is approximately quadratic in the number of examples, this way we can considerably accelerate the process of estimating support vector weights.

According to statistical learning theory, being trained on smaller subsets of the available data these models will be less robust with respect to the

minimization of the empirical risk (Vapnik, 1998). Nonetheless, since we do not need to employ them for classification (but just to direct our feature selection process, as we will describe shortly), we can accept to rely on sub-optimal weights. Furthermore, research results in the field of SVM parallelization using cascades of SVMs (Graf et al., 2004) suggest that support vectors collected from locally learnt models can encode many of the relevant features retained by models learnt globally. Henceforth, let M_s be the model associated with the s-th split, and $\mathcal{F}_s$ the fragment space that can describe all the trees in M_s.

Fragment Mining and Indexing (*FMI*) In Equation 1 it is possible to isolate the gradient $\vec{w} = \sum_{i=1}^{n} \alpha_i y_i \vec{x_i}$, with $\vec{x_i} = [x_i^{(1)}, \ldots, x_i^{(N)}]$, N being the dimensionality of the feature space. For a tree kernel function, we can rewrite $x_i^{(j)}$ as:

$$x_i^{(j)} = \frac{t_{i,j}\lambda^{\ell(f_j)}}{\|t_i\|} = \frac{t_{i,j}\lambda^{\ell(f_j)}}{\sqrt{\sum_{k=1}^{N}(t_{i,k}\lambda^{\ell(f_k)})^2}} \qquad (2)$$

where: $t_{i,j}$ is the number of occurrences of the fragment f_j, associated with the j-th dimension of the feature space, in the tree t_i; λ is the kernel decay factor; and $\ell(f_j)$ is the depth of the fragment.

The relevance $|w^{(j)}|$ of the fragment f_j can be measured as:

$$|w^{(j)}| = \left| \sum_{i=1}^{n} \alpha_i y_i x_i^{(j)} \right| . \qquad (3)$$

We fix a threshold L and from each model M_s (learnt during FSL) we select the L most relevant fragments, i.e. we build the set $\mathcal{F}_{s,L} = \cup_k \{f_k\}$ so that:

$$|\mathcal{F}_{s,L}| = L \text{ and } |w^{(k)}| \geq |w^{(i)}| \forall f_i \in \mathcal{F} \setminus \mathcal{F}_{s,L} .$$

In order to do so, we need to harvest all the fragments with a fast extraction function, store them in a compact data structure and finally select the fragments with the highest relevance. Our strategy is exemplified in Figure 2. First, we represent each fragment as a sequence as described in (Zaki, 2002). A sequence contains the labels of the fragment nodes in depth-first order. By default, each node is the child of the previous node in the sequence. A special symbol (↑) indicates that the next node in the sequence should be attached after climbing one level in the tree. For example, the tree *(B (Z W))* in figure is represented as the sequence *[B, Z, ↑, W]*. Then, we add the elements of the sequence to a graph (which we call an *index* of fragments) where each sequence becomes a path. The nodes of the index are the labels of the fragment nodes, and each arc is associated with a pair of values $\langle d, n \rangle$: d is a node identifier, which is unique with respect to the source node; n is the identifier of the arc that must be selected at the destination node in order to follow the path associated with the sequence. Index nodes associated with a fragment root also have a field where the cumulative relevance of the fragment is stored.

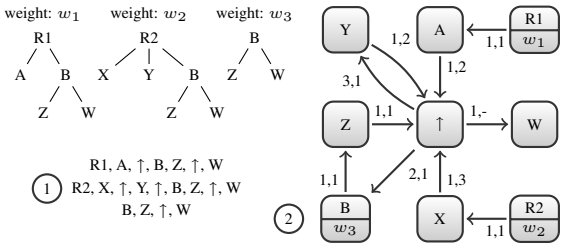


Figure 2: Fragment indexing. Each fragment is represented as a sequence ① and then encoded as a path in the index ② which keeps track of its cumulative relevance.

As an example, the index node labeled *B* in figure has an associated weight of w_3, thus identifying the root of a fragment. Each outgoing edge univocally identifies an indexed fragment. In this case, the only outgoing edge is labeled with the pair $\langle d = 1, n = 1 \rangle$, meaning that we should follow it to the next node, i.e. *Z*, and there select the edge labeled *1*, as indicated by n. The edge with $d = 1$ in *Z* is $\langle d = 1, n = 1 \rangle$, so we browse to ↑ where we select the edge $\langle d = 1, n = - \rangle$. The missing value for n tells us that the next node, *W*, is the last element of the sequence. The complete sequence is then *[B, Z, ↑, W]*, which encodes the fragment *(B (Z W))*.

The index implementation has been optimized for fast insertions and has the following features: 1) each node label is represented exactly once; 2) each distinct sequence tail is represented exactly once. The union of all the fragments harvested from each model is then saved into a dictionary $\mathcal{D}_L$ which will be used by the next stage.

To mine the fragments, we apply to each tree in each model the algorithm shown in Algorithm 3.1. In this context, we call *fragment expansion* the pro-

Algorithm 3.1: MINE_TREE($tree$)

global $maxdepth, maxexp$
main
$mined \leftarrow \emptyset; indexed \leftarrow \emptyset;$ MINE(FRAG($tree$), 0)
procedure MINE($frag, depth$)
if $frag \in indexed$
then return
$indexed \leftarrow indexed \cup \{frag\}$
INDEX($frag$)
for each $node \in$ TO_EXPAND($frag$)
do { **if** $node \notin mined$
then { $mined \leftarrow mined \cup \{node\}$
MINE(FRAG($node$), 0)
if $depth < maxdepth$
then { **for each** $fragment \in$ EXPAND($frag, maxexp$)
do MINE($fragment, depth + 1$)

cess by which tree nodes are included in a fragment. Fragment expansion is achieved via *node expansions*, where expanding a node means including its direct children in the fragment. The function FRAG(n) builds the basic fragment rooted in a given node n, i.e. the fragment consisting only of n and its direct children. The function TO_EXPAND(f) returns the set of nodes in a fragment f that can be expanded (i.e. internal nodes in the origin tree), while the function EXPAND($f, maxexp$) returns all the possible expansions of a fragment f. The parameter $maxexp$ is a limit to the number of nodes that can be expanded at the same time when a new fragment is generated, while $maxdepth$ sets a limit on the number of times that a base fragment can be expanded. The function INDEX(f) adds the fragment f to the index. To keep the notation simple, here we assume that a fragment f contains all the necessary information to calculate its relevance (i.e. the weight, label and norm of the support vector α_i, y_i, and $\|t_i\|$, the depth of the fragment $\ell(f)$ and the decay factor λ, see equations 2 and 3).

Performing in a different order the same node expansions on the same fragment f results in the same fragment f'. To prevent the algorithm from entering circular loops, we use the set $indexed$ so that the very same fragment in each tree cannot be explored more than once. Similarly, the $mined$ set is used so that the base fragment rooted in a given node is considered only once.

Tree Fragment Extraction (*TFX*) During this phase, a data file encoding label-tree pairs $\langle y_i, t_i \rangle$ is

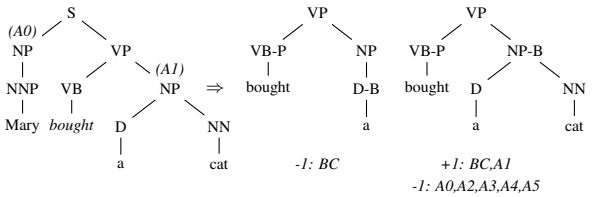


Figure 3: Examples of AST_m structured features.

transformed to encode label-vector pairs $\langle y_i, \vec{v_i} \rangle$. To do so, we generate the fragment space of t_i, using a variant of the mining algorithm described in Figure 3.1, and encode in $\vec{v_i}$ all and only the fragments $t_{i,j}$ so that $t_{i,j} \in \mathcal{D}_L$, i.e. we perform feature extraction based on the indexed fragments. The process is applied to the whole training and test sets. The algorithm exploits labels and production rules found in the fragments listed in the dictionary to generate only the fragments that *may be* in the dictionary. For example, if the dictionary does not contain a fragment whose root is labeled N, then if a node N is encountered during TFX neither its base fragment nor its expansions are generated.

Explicit Space Learning (*ESL*) After linearizing the training data, we can learn a very fast model by using all the available data and a linear kernel. The fragment space is now *explicit*, as there is a mapping between the input vectors and the fragments they encode.

Explicit Space Classification (*ESC*) After learning the linear model, we can classify our linearized test data and evaluate the accuracy of the resulting classifier.

4 Previous work

A rather comprehensive overview of feature selection techniques is carried out in (Guyon and Elisseeff, 2003). Non-filter approaches for SVMs and kernel machines are often concerned with polynomial and Gaussian kernels, e.g. (Weston et al., 2001) and (Neumann et al., 2005). Weston et al. (2003) use the ℓ_0 norm in the SVM optimizer to stress the feature selection capabilities of the learning algorithm. In (Kudo and Matsumoto, 2003), an extension of the PrefixSpan algorithm (Pei et al., 2001) is used to efficiently mine the features in a low degree polynomial kernel space. The authors discuss an approximation of their method that allows them to handle high degree polynomial kernels.

Data set			Non-linearized classifiers					Linearized classifiers (Thr=10k)				
Task	Pos	Neg	Train	Test	P	R	F_1	Train	Test	P	R	F_1
A0	60,900	118,191	521	7	90.26	92.95	91.59	209	3	88.95	91.91	90.40
A1	90,636	88,455	1,206	11	89.45	88.62	89.03	376	3	89.39	88.13	88.76
A2	21,291	157,800	692	7	84.56	64.42	73.13	248	3	81.23	68.29	74.20
A3	3,481	175,610	127	2	97.67	40.00	56.76	114	3	97.56	38.10	54.79
A4	2,713	176,378	47	1	92.68	55.07	69.10	92	2	95.00	55.07	69.72
A5	69	179,022	3	0	100.00	50.00	66.67	63	2	100.00	50.00	66.67
BC	61,062	938,938	3,059	247	82.57	80.96	81.76	916	39	83.36	78.95	81.10
RM	-	-	2,596	27	89.37	86.00	87.65	1,090	16	88.50	85.81	87.13

Table 1: Accuracy (*P*, *R*, F_1), training (*Train*) and test (*Test*) time of non-linearized (center) and linearized (right) classifiers. Times are in minutes. For each task, columns *Pos* and *Neg* list the number of positive and negative training examples, respectively. The accuracy of the role multiclassifiers is the micro-average of the individual classifiers trained to recognize core PropBank roles.

Suzuki and Isozaki (2005) present an embedded approach to feature selection for convolution kernels based on χ^2-driven relevance assessment. To our knowledge, this is the only published work clearly focusing on feature selection for tree kernel functions. In (Graf et al., 2004), an approach to SVM parallelization is presented which is based on a divide-et-impera strategy to reduce optimization time. The idea of using a compact graph representation to represent the support vectors of a TK function is explored in (Aiolli et al., 2006), where a Direct Acyclic Graph (DAG) is employed.

Concerning the use of kernels for NLP, interesting models and results are described, for example, in (Collins and Duffy, 2002), (Moschitti et al., 2008), (Kudo and Matsumoto, 2003), (Cumby and Roth, 2003), (Shen et al., 2003), (Cancedda et al., 2003), (Culotta and Sorensen, 2004), (Daumé III and Marcu, 2004), (Kazama and Torisawa, 2005), (Kudo et al., 2005), (Titov and Henderson, 2006), (Moschitti et al., 2006), (Moschitti and Bejan, 2004) or (Toutanova et al., 2004).

5 Experiments

We tested our model on a Semantic Role Labeling (SRL) benchmark, using PropBank annotations (Palmer et al., 2005) and automatic Charniak parse trees (Charniak, 2000) as provided for the CoNLL 2005 evaluation campaign (Carreras and Màrquez, 2005). SRL can be decomposed into two tasks: *boundary detection*, where the word sequences that are arguments of a predicate word w are identified, and *role classification*, where each argument is assigned the proper role. The former task requires a binary *Boundary Classifier* (BC), whereas the second involves a *Role Multi-class Classifier* (RM).

Setup. If the constituency parse tree t of a sentence s is available, we can look at all the pairs $\langle p, n_i \rangle$, where n_i is any node in the tree and p is the node dominating w, and decide whether n_i is an *argument node* or not, i.e. whether it exactly dominates all and only the words encoding any of w's arguments. The objects that we classify are subsets of the input parse tree that encompass both p and n_i. Namely, we use the AST_m structure defined in (Moschitti et al., 2008), which is the minimal tree that covers all and only the words of p and n_i. In the AST_m, p and n_i are marked so that they can be distinguished from the other nodes. An AST_m is regarded as a positive example for BC if n_i is an argument node, otherwise it is considered a negative example. Positive BC examples can be used to train an efficient RM: for each role r we can train a classifier whose positive examples are argument nodes whose label is exactly r, whereas negative examples are argument nodes labeled $r' \neq r$. Two AST_ms extracted from an example parse tree are shown in Figure 3: the first structure is a negative example for BC and is not part of the data set of RM, whereas the second is a positive instance for BC and A1.

To train BC we used PropBank sections 1 through 6, extracting AST_m structures out of the first 1 million $\langle p, n_i \rangle$ pairs from the corresponding parse trees. As a test set we used the 149,140 instance collected from the annotations in Section 24. There are 61,062 positive examples in the training set (i.e. 6.1%) and 8,515 in the test set (i.e. 5.7%).

For RM we considered all the argument nodes of any of the six PropBank core roles (i.e. A0, ...,

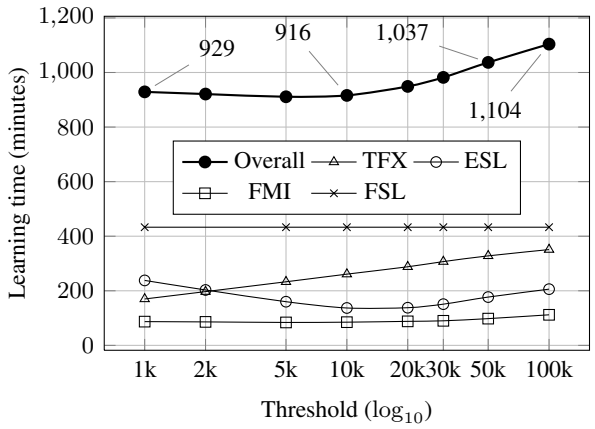


Figure 4: Training time decomposition for the linearized BC with respect to its main components when varying the threshold value.

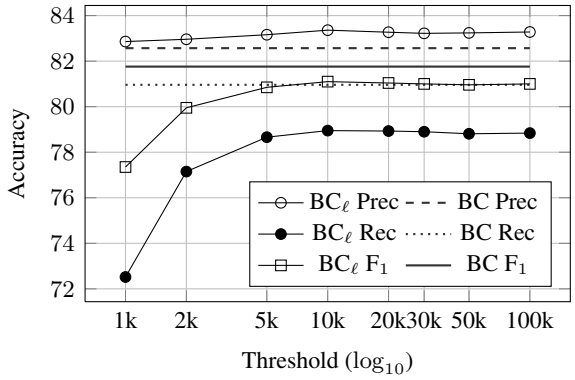


Figure 5: BC$_\ell$ accuracy for different thresholds.

A5) from all the available training sections, i.e. 2 through 21, for a total of 179,091 training instances. Similarly, we collected 5,928 test instances from the annotations of Section 24.

In the remainder, we will mark with an ℓ the linearized classifiers, i.e. BC$_\ell$ and RM$_\ell$ will refer to the linearized boundary and role classifiers, respectively. Their traditional, vanilla SST counterparts will be simply referred to as BC and RM.

We used 10 splits for the FMI stage and we set $maxdepth = 4$ and $maxexp = 5$ during FMI and TFX. We didn't carry out an extensive validation of these parameters. These values were selected during the development of the software because, on a very small development set, they resulted in a very responsive system.

Since the main topic of this paper is the assessment of the efficiency and accuracy of our linearization technique, we did not carry out an evaluation on the whole SRL task using the official CoNLL'05 evaluator. Indeed, producing complete annotations requires several steps (e.g. overlap resolution, OvA or Pairwise combination of individual role classifiers) that would shade off the actual impact of the methodology on classification.

Platform. All the experiments were run on a machine equipped with 4 Intel® Xeon® CPUs clocked at 1.6 GHz and 4 GB of RAM running on a Linux 2.6.9 kernel. As a supervised learning framework we used SVM-Light-TK [1], which extends the SVM-Light optimizer (Joachims, 2000) with tree kernel support. During FSL, we learn the models using a normalized SST kernel and the default decay factor $\lambda = 0.4$. The same parameters are used to train the models of the non linearized classifiers. During ESL, the classifier is trained using a linear kernel. We did not carry out further parametrization of the learning algorithm.

Results. The left side of Table 1 shows the distribution of positive (Column *Pos*) and negative (*Neg*) data points in each classifier's training set. The central group of columns lists training and test efficiency and accuracy of BC and RM, i.e. the non-linearized classifiers, along with figures for the individual role classifiers that make up RM.

Training BC took more than two days of CPU time and testing about 4 hours. The classifier achieves an F_1 measure of 81.76, with a good balance between precision and recall. Concerning RM, sequential training of the 6 models took 2,596 minutes, while classification took 27 minutes. The slowest of the individual role classifiers happens to be A1, which has an almost 1:1 ratio between positive and negative examples, i.e. they are 90,636 and 88,455 respectively.

We varied the threshold value (i.e. the number of fragments that we mine from each model, see Section 3) to measure its effect on the resulting classifier accuracy and efficiency. In this context, we call *training time* all the time necessary to obtain a linearized model, i.e. the sum of FSL, FMI and TFX time for every split, plus the time for ESL. Similarly, we call *test time* the time necessary to classify a linearized test set, i.e. the sum of TFX and ESC on test data.

In Figure 4 we plot the efficiency of BC$_\ell$ learn-

[1] http://disi.unitn.it/~moschitt/Tree-Kernel.htm

ing with respect to different threshold values. The Overall training time is shown alongside with partial times coming from FSL (which is the same for every threshold value and amounts to 433 minutes), FMI, training data TFX and ESL. The plot shows that TFX has a logarithmic behaviour, and that quite soon becomes the main player in total training time after FSL. For threshold values lower than 10k, ESL time decreases as the threshold increases: too few fragments are available and adding new ones increases the probability of including relevant fragments in the dictionary. After 10k, all the relevant fragments are already there and adding more only makes computation harder. We can see that for a threshold value of 100k total training time amounts to 1,104 minutes, i.e. 36% of BC. For a threshold value of 10k, learning time further decreases to 916 minutes, i.e. less than 30%. This threshold value was used to train the individual linearized role classifiers that make up RM_ℓ.

These considerations are backed by the trend of classification accuracy shown in Figure 5, where the Precision, Recall and F_1 measure of BC_ℓ, evaluated on the test set, are shown in comparison with BC. We can see that BC_ℓ precision is almost constant, while its recall increases as we increase the threshold, reaches a maximum of 78.95% for a threshold of 10k and then settles around 78.8%. The F_1 score is maximized for a threshold of 10k, where it measures 81.10, i.e. just 0.66 points less than BC. We can also see that BC_ℓ is constantly more conservative than BC, i.e. it always has higher precision and lower recall.

Table 1 compares side to side the accuracy (columns *P*, *R* and F_1), training (*Train*) and test (*Test*) times of the different classifiers (central block of columns) and their linearized counterparts (block on the right). Times are measured in minutes. For the linearized classifiers, test time is the sum of TFX and ESC time, but the only relevant contribution comes from TFX, as the low dimensional linear space and fast linear kernel allow us to classify test instances very efficiently [2]. Overall, BC_ℓ test time is 39 minutes, which is more than 6 times faster than BC (i.e. 247 minutes). It should be stressed that we are comparing against a fast TK implementation that is almost linear in time with respect to the number of tree nodes (Moschitti, 2006).

[2] Although ESC is not shown in table, the classification of all 149k test instances with BC_ℓ took 5 seconds with a threshold of 1k and 17 seconds with a threshold of 100k.

Learning parallelization

Task	Non Lin.	Linearized (Thr=10k)		
		1 *cpu*	5 *cpus*	10 *cpus*
BC	3,059	916	293	215
RM	2,596	1,090	297	198

Table 2: Learning time when exploiting the framework's parallelization capabilities. Column *Non Lin.* lists non-linearized training time.

Concerning RM_ℓ, we can see that the accuracy loss is even less than with BC_ℓ, i.e. it reaches an F_1 measure of 87.13 which is just 0.52 less than RM. It is also interesting to note how the individual linearized role classifiers manage to perform accurately regardless of the distribution of examples in the data set: for all the six classifiers the final accuracy is in line with that of the corresponding non-linearized classifier. In two cases, i.e. A2 and A4, the accuracy of the linearized classifier is even higher, i.e. 74.20 vs. 73.13 and 69.72 vs. 69.10, respectively. As for the efficiency, total training time for RM_ℓ is 37% of RM, i.e. 1,190 vs. 2,596 minutes, while test time is reduced to 60%, i.e. 16 vs 27 minutes. These improvements are less evident than those measured for boundary detection. The main reason is that the training set for boundary classification is much larger, i.e. 1 million vs. 179k instances: therefore, splitting training data during FSL has a reduced impact on the overall efficiency of RM_ℓ.

Parallelization. All the efficiency improvements that have been discussed so far considered a completely sequential process. But one of the advantages of our approach is that it allows us to parallelize some aspect of SVM training. Indeed, every activity (but ESL) can exploit some degree of parallelism: during FSL, all the models can be learnt at the same time (for this activity, the maximum degree of parallelization is conditioned by the number of training data splits); during FMI, models can be mined concurrently; during TFX, the data-set to be linearized can be split arbitrarily and individual segments can be processed in parallel. Exploiting this possibility we can drastically improve learning efficiency. As an example, in Table 2 we show how the total learning of the BC_ℓ can be cut to as low as 215 seconds when exploiting ten CPUs and using a

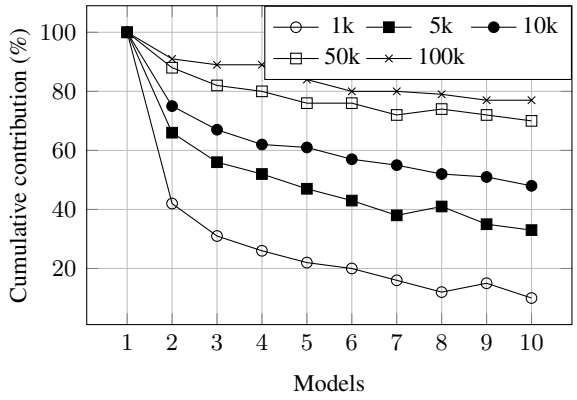


Figure 6: Growth of dictionary size when including fragments from more splits at different threshold values. When a low threshold is used, the contribution of individual dictionaries tends to be more marginal.

threshold of 10k. Even running on just 5 CPUs, the overall computational cost of BC_ℓ is less than 10% of BC (Column *Non Lin.*). Similar considerations can be drawn concerning the role multi-classifier.

Fragment space. In this section we take a look at the fragments included in the dictionary of the BC_ℓ classifier. During FMI, we incrementally merge the fragments mined from each of the models learnt during FSL. Figure 6 plots, for different threshold values, the percentage of new fragments (on the y axis) that the i-th model (on the x axis) contributes with respect to the number of fragments mined from each model (i.e. the threshold value).

If we consider the curve for a threshold equal to 100k, we can see that each model after the first approximately contributes with the same number of fragments. On the other hand, if the threshold is set to 1k than the contribution of subsequent models is increasingly more marginal. Eventually, less than 10% of the fragments mined from the last model are new ones. This behaviour suggests that there is a core set of very relevant fragments which is common across models learnt on different data, i.e. they are relevant for the task and do not strictly depend on the training data that we use. When we increase the threshold value, the new fragments that we index are more and more data specific.

The dictionary compiled with a threshold of 10k lists 62,760 distinct fragments. 15% of the fragments contain the predicate node (which generally is the node encoding the predicate word's POS tag), more than one third contain the candidate argument node and, of these, about one third are rooted in it. This last figure strongly suggests that the internal structure of an argument is indeed a very powerful feature not only for role classification, as we would expect, but also for boundary detection. About 10% of the fragments contain both the predicate and the argument node, while about 1% encode the Path feature traditionally used in explicit semantic role labeling models (Gildea and Jurafsky, 2002). About 5% encode a sort of extended Path feature, where the argument node is represented together with its descendants. Overall, about 2/3 of the fragments contain at least some terminal symbol (i.e. words), generally a preposition or an adverb.

6 Conclusions

We presented a supervised learning framework for Support Vector Machines that tries to combine the power and modeling simplicity of convolution kernels with the advantages of linear kernels and explicit feature representations. We tested our model on a Semantic Role Labeling benchmark and obtained very promising results in terms of accuracy and efficiency. Indeed, our linearized classifiers manage to be almost as accurate as non linearized ones, while drastically reducing the time required to train and test a model on the same amounts of data.

To our best knowledge, the main points of novelty of this work are the following: 1) it addresses the problem of feature selection for tree kernels, exploiting SVM decisions to guide the process; 2) it provides an effective way to make the kernel space observable; 3) it can efficiently linearize structured data without the need for an explicit mapping; 4) it combines feature selection and SVM parallelization.

We began investigating the fragments generated by a TK function for SRL, and believe that studying them in more depth will be useful to identify new, relevant features for the characterization of predicate-argument relations.

In the months to come, we plan to run a set of experiments on a wider list of tasks so as to consolidate the results we obtained so far. We will also test the generality of the approach by testing with different high-dimensional kernel families, such as sequence and polynomial kernels.

References

Fabio Aiolli, Giovanni Da San Martino, Alessandro Sperduti, and Alessandro Moschitti. 2006. Fast on-line kernel learning for trees. In *Proceedings of ICDM'06*.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. 2003. Word sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL'05*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL'00*.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of ACL'02*.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of ACL'04*.

Chad Cumby and Dan Roth. 2003. Kernel Methods for Relational Learning. In *Proceedings of ICML 2003*.

Hal Daumé III and Daniel Marcu. 2004. Np bracketing by maximum entropy tagging and SVM reranking. In *Proceedings of EMNLP'04*.

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

Hans P. Graf, Eric Cosatto, Leon Bottou, Igor Durdanovic, and Vladimir Vapnik. 2004. Parallel support vector machines: The cascade svm. In *Neural Information Processing Systems*.

Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

David Haussler. 1999. Convolution kernels on discrete structures. Technical report, Dept. of Computer Science, University of California at Santa Cruz.

T. Joachims. 2000. Estimating the generalization performance of a SVM efficiently. In *Proceedings of ICML'00*.

Hisashi Kashima and Teruo Koyanagi. 2002. Kernels for semi-structured data. In *Proceedings of ICML'02*.

Jun'ichi Kazama and Kentaro Torisawa. 2005. Speeding up training with tree kernels for node relation labeling. In *Proceedings of HLT-EMNLP'05*.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of ACL'03*.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proceedings of ACL'05*.

Alessandro Moschitti and Cosmin Bejan. 2004. A semantic kernel for predicate argument classification. In *CoNLL-2004*, Boston, MA, USA.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2006. Semantic role labeling via tree kernel joint inference. In *Proceedings of CoNLL-X*, New York City.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning. In *Proccedings of EACL'06*.

Julia Neumann, Christoph Schnorr, and Gabriele Steidl. 2005. Combined SVM-Based Feature Selection and Classification. *Machine Learning*, 61(1-3):129–150.

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

J. Pei, J. Han, Mortazavi B. Asl, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu. 2001. PrefixSpan Mining Sequential Patterns Efficiently by Prefix Projected Pattern Growth. In *Proceedings of ICDE'01*.

Alain Rakotomamonjy. 2003. Variable selection using SVM based criteria. *Journal of Machine Learning Research*, 3:1357–1370.

Libin Shen, Anoop Sarkar, and Aravind k. Joshi. 2003. Using LTAG Based Features in Parse Reranking. In *Proceedings of EMNLP'06*.

Jun Suzuki and Hideki Isozaki. 2005. Sequence and Tree Kernels with Statistical Feature Mining. In *Proceedings of the 19th Annual Conference on Neural Information Processing Systems (NIPS'05)*.

Ivan Titov and James Henderson. 2006. Porting statistical parsers with data-defined kernels. In *Proceedings of CoNLL-X*.

Kristina Toutanova, Penka Markova, and Christopher Manning. 2004. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In *Proceedings of EMNLP 2004*.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio, and Vladimir Vapnik. 2001. Feature Selection for SVMs. In *Proceedings of NIPS'01*.

Jason Weston, André Elisseeff, Bernhard Schölkopf, and Mike Tipping. 2003. Use of the zero norm with linear models and kernel methods. *J. Mach. Learn. Res.*, 3:1439–1461.

Mohammed J Zaki. 2002. Efficiently mining frequent trees in a forest. In *Proceedings of KDD'02*.

A Method for Stopping Active Learning Based on Stabilizing Predictions and the Need for User-Adjustable Stopping

Michael Bloodgood*
Human Language Technology
Center of Excellence
Johns Hopkins University
Baltimore, MD 21211 USA
bloodgood@jhu.edu

K. Vijay-Shanker
Computer and Information
Sciences Department
University of Delaware
Newark, DE 19716 USA
vijay@cis.udel.edu

Abstract

A survey of existing methods for stopping active learning (AL) reveals the needs for methods that are: more widely applicable; more aggressive in saving annotations; and more stable across changing datasets. A new method for stopping AL based on stabilizing predictions is presented that addresses these needs. Furthermore, stopping methods are required to handle a broad range of different annotation/performance tradeoff valuations. Despite this, the existing body of work is dominated by conservative methods with little (if any) attention paid to providing users with control over the behavior of stopping methods. The proposed method is shown to fill a gap in the level of aggressiveness available for stopping AL and supports providing users with control over stopping behavior.

1 Introduction

The use of Active Learning (AL) to reduce NLP annotation costs has generated considerable interest recently (e.g. (Bloodgood and Vijay-Shanker, 2009; Baldridge and Osborne, 2008; Zhu et al., 2008a)). To realize the savings in annotation efforts that AL enables, we must have a mechanism for knowing when to stop the annotation process.

Figure 1 is intended to motivate the value of stopping at the right time. The x-axis measures the number of human annotations that have been requested and ranges from 0 to 70,000. The y-axis measures performance in terms of F-Measure. As can be seen from the figure, the issue is that if we stop too early while useful generalizations are still being made, we wind up with a lower performing system but if we stop too late after all the useful generalizations have been made, we just wind up wasting human annotation effort.

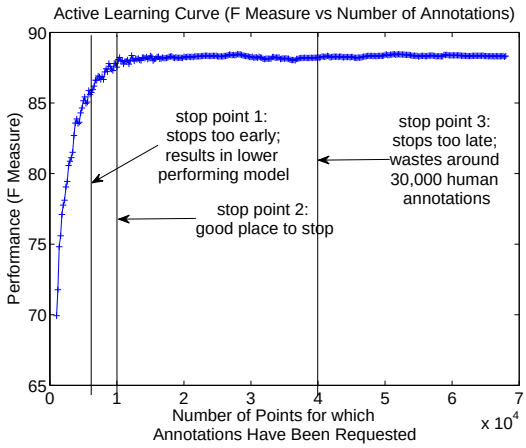


Figure 1: Hypothetical Active Learning Curve with hypothetical stopping points.

The terms *aggressive* and *conservative* will be used throughout the rest of this paper to describe the behavior of stopping methods. Conservative methods tend to stop further to the right in Figure 1. They are conservative in the sense that they're very careful not to risk losing significant amounts of F-measure, even if it means annotating many more examples than necessary. Aggressive methods, on the other hand, tend to stop further to the left in Figure 1. They are aggressively trying to reduce unnecessary annotations.

There has been a flurry of recent work tackling the

* This research was conducted while the first author was a PhD student at the University of Delaware.

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 39–47,
Boulder, Colorado, June 2009.

problem of automatically determining when to stop AL (see Section 2). There are three areas where this body of work can be improved:

applicability Several of the leading methods are restricted to only being used in certain situations, e.g., they can't be used with some base learners, they have to select points in certain batch sizes during AL, etc. (See Section 2 for discussion of the exact applicability constraints of existing methods.)

lack of aggressive stopping The leading methods tend to find stop points that are too far to the right in Figure 1. (See Section 4 for empirical confirmation of this.)

instability Some of the leading methods work well on some datasets but then can completely break down on other datasets, either stopping way too late and wasting enormous amounts of annotation effort or stopping way too early and losing large amounts of F-measure. (See Section 4 for empirical confirmation of this.)

This paper presents a new stopping method based on stabilizing predictions that addresses each of these areas and provides user-adjustable stopping behavior. The essential idea behind the new method is to test the predictions of the recently learned models (during AL) on examples which don't have to be labeled and stop when the predictions have stabilized. Some of the main advantages of the new method are that: it requires no additional labeled data, it's widely applicable, it fills a need for a method which can aggressively save annotations, it has stable performance, and it provides users with control over how aggressively/conservatively to stop AL.

Section 2 discusses related work. Section 3 explains our Stabilizing Predictions (SP) stopping criterion in detail. Section 4 evaluates the SP method and discusses results. Section 5 concludes.

2 Related Work

Laws and Schütze (2008) present stopping criteria based on the gradient of performance estimates and the gradient of confidence estimates. Their technique with gradient of performance estimates is only applicable when probabilistic base learners are used. The gradient of confidence estimates method is more generally applicable (e.g., it can be applied with our experiments where we use SVMs as the base learner). This method, denoted by LS2008 in Tables and Figures, measures the rate of change of model confidence over a window of recent points and when the gradient falls below a threshold, AL is stopped.

The margin exhaustion stopping criterion was developed for AL with SVMs (AL-SVM). It says to stop when all of the remaining unlabeled examples are outside of the current model's margin (Schohn and Cohn, 2000) and is denoted as SC2000 in Tables and Figures. Ertekin et al. (2007) developed a similar technique that stops when the number of support vectors saturates. This is equivalent to margin exhaustion in all of our experiments so this method is not shown explicitly in Tables and Figures. Since we use AL with SVMs, we will compare with margin exhaustion in our evaluation section. Unlike our SP method, margin exhaustion is only applicable for use with margin-based methods such as SVMs and can't be used with other base learners such as Maximum Entropy, Naive Bayes, and others. Schohn and Cohn (2000) show in their experiments that margin exhaustion has a tendency to stop late. This is further confirmed in our experiments in Section 4.

The confidence-based stopping criterion (hereafter, V2008) in (Vlachos, 2008) says to stop when model confidence consistently drops. As pointed out by (Vlachos, 2008), this stopping criterion is based on the assumption that the learner/feature representation is incapable of fully explaining all the examples. However, this assumption is often violated and then the performance of the method suffers (see Section 4).

Two stopping criteria (max-conf and min-err) are reported in (Zhu and Hovy, 2007). The max-conf method indicates to stop when the confidence of the model on each unlabeled example exceeds a threshold. In the context of margin-based methods, max-conf boils down to be simply a generalization of the margin exhaustion method. Min-err, reported to be superior to max-conf, says to stop when the accuracy of the most recent model on the current batch of queried examples exceeds some threshold (they use 0.9). Zhu et al. (2008b) proposes the use of multi-criteria-based stopping to handle setting the thresh-

old for min-err. They refuse to stop and they raise the min-err threshold if there have been any classification changes on the remaining unlabeled data by consecutive actively learned models when the current min-err threshold is satisfied. We denote this multi-criteria-based strategy, reported to work better than min-err in isolation, by ZWH2008. As seen in (Zhu et al., 2008a), sometimes min-err indeed stops later than desired and ZWH2008 must (by nature of how it operates) stop at least as late as min-err does. The susceptibility of ZWH2008 to stopping late is further shown empirically in Section 4. Also, ZWH2008 is not applicable for use with AL setups that select examples in small batches.

3 A Method for Stopping Active Learning Based on Stabilizing Predictions

To stop active learning at the point when annotations stop providing increases in performance, perhaps the most straightforward way is to use a separate set of labeled data and stop when performance begins to level off on that set. But the problem with this is that it requires additional labeled data which is counter to our original reason for using AL in the first place. Our hypothesis is that we can sense when to stop AL by looking at (only) the *predictions* of consecutively learned models on examples that don't have to be labeled. We won't know if the predictions are correct or not but we can see if they have stabilized. If the predictions have stabilized, we hypothesize that the performance of the models will have stabilized *and vice-versa*, which will ensure a (much-needed) aggressive approach to saving annotations.

SP checks for stabilization of predictions on a set of examples, called the stop set, that don't have to be labeled. Since stabilizing predictions on the stop set is going to be used as an indication that model stabilization has occurred, the stop set ought to be representative of the types of examples that will be encountered at application time. There are two conflicting factors in deciding upon the size of the stop set to use. On the one hand, a small set is desirable because then SP can be checked quickly. On the other hand, a large set is desired to ensure we don't make a decision based on a set that isn't representative of the application space. As a compromise between these factors, we chose a size of 2000. In Section 4, sensitivity analysis to stop set size is performed and more principled methods for determining stop set size and makeup are discussed.

It's important to allow the examples in the stop set to be queried if the active learner selects them because they may be highly informative and ruling them out could hurt performance. In preliminary experiments we had made the stop set distinct from the set of unlabeled points made available for querying and we saw performance was *qualitatively* the same but the AL curve was translated down by a few F-measure points. Therefore, we allow the points in the stop set to be selected during AL.[1]

The essential idea is to compare successive models' predictions on the stop set to see if they have stabilized. A simple way to define agreement between two models would be to measure the percentage of points on which the models make the same predictions. However, experimental results on a separate development dataset show then that the cutoff agreement at which to stop is sensitive to the dataset being used. This is because different datasets have different levels of agreement that can be expected by chance and simple percent agreement doesn't adjust for this.

Measurement of agreement between human annotators has received significant attention and in that context, the drawbacks of using percent agreement have been recognized (Artstein and Poesio, 2008). Alternative metrics have been proposed that take chance agreement into account. In (Artstein and Poesio, 2008), a survey of several agreement metrics is presented. Most of the agreement metrics are of the form:

$$agreement = \frac{A_o - A_e}{1 - A_e}, \qquad (1)$$

where A_o = observed agreement, and A_e = agreement expected by chance. The different metrics differ in how they compute A_e.

The Kappa statistic (Cohen, 1960) measures agreement expected by chance by modeling each coder (in our case model) with a separate distribution governing their likelihood of assigning a particular category. Formally, Kappa is defined by Equa-

[1] They remain in the stop set if they're selected.

tion 1 with A_e computed as follows:

$$A_e = \sum_{k \in \{+1,-1\}} P(k|c_1) \cdot P(k|c_2), \qquad (2)$$

where each c_i is one of the coders (in our case, models), and $P(k|c_i)$ is the probability that coder (model) c_i labels an instance as being in category k. Kappa estimates $P(k|c_i)$ based on the proportion of observed instances that coder (model) c_i labeled as being in category k.

We have found Kappa to be a robust parameter that doesn't require tuning when moving to a new dataset. On a separate development dataset, a Kappa cutoff of 0.99 worked well. All of the experiments (except those in Table 2) in the current paper used an agreement cutoff of Kappa = 0.99 with zero tuning performed. We will see in Section 4 that this cutoff delivers robust results across all of the folds for all of the datasets.

The Kappa cutoff captures the *intensity* of the agreement that must occur before SP will conclude to stop. Though an intensity cutoff of K=0.99 is an excellent default (as seen by the results in Section 4), one of the advantages of the SP method is that by giving users the option to vary the intensity cutoff, users can control how aggressive SP will behave. This is explored further in Section 4.

Another way to give users control over stopping behavior is to give them control over the *longevity* for which agreement (at the specified intensity) must be maintained before SP concludes to stop. The simplest implementation would be to check the most recent model with the previous model and stop if their agreement exceeds the intensity cutoff. However, independent of wanting to provide users with a longevity control, this is not an ideal approach because there's a risk that these two models could happen to highly agree but then the next model will not highly agree with them. Therefore, we propose using the average of the agreements from a window of the k most recent pairs of models. If we call the most recent model M_n, the previous model M_{n-1} and so on, with a window size of 3, we average the agreements between M_n and M_{n-1}, between M_{n-1} and M_{n-2}, and between M_{n-2} and M_{n-3}. On separate development data a window size of k=3 worked well. All of the experiments (except those in Table 3) in the current paper used a longevity window size of k=3 with zero tuning performed. We will see in Section 4 that this longevity default delivers robust results across all of the folds for all of the datasets. Furthermore, Section 4 shows that varying the longevity requirement provides users with another lever for controlling how aggressively SP will behave.

4 Evaluation and Discussion

4.1 Experimental Setup

We evaluate the Stabilizing Predictions (SP) stopping method on multiple datasets for Text Classification (TC) and Named Entity Recognition (NER) tasks. All of the datasets are freely and publicly available and have been used in many past works.

For Text Classification, we use two publicly available spam corpora: the spamassassin corpus used in (Sculley, 2007) and the TREC spam corpus trec05p-1/ham25 described in (Cormack and Lynam, 2005). For both of these corpora, the task is a binary classification task and we perform 10-fold cross validation. We also use the Reuters dataset, in particular the Reuters-21578 Distribution 1.0 ModApte split[2]. Since a document may belong to more than one category, each category is treated as a separate binary classification problem, as in (Joachims, 1998; Dumais et al., 1998). Consistent with (Joachims, 1998; Dumais et al., 1998), results are reported for the ten largest categories. Other TC datasets we use are the 20Newsgroups[3] newsgroup article classification and the WebKB web page classification datasets. For WebKB, as in (McCallum and Nigam, 1998; Zhu et al., 2008a; Zhu et al., 2008b) we use the four largest categories. For all of our TC datasets, we use binary features for every word that occurs in the training data at least three times.

For NER, we use the publicly available GENIA corpus[4]. Our features, based on those from (Lee et al., 2004), are surface features such as the words in

[2]http://www.daviddlewis.com/resources/testcollections/reuters21578

[3]We used the "bydate" version of the dataset downloaded from http://people.csail.mit.edu/jrennie/20Newsgroups/. This version is recommended since it makes cross-experiment comparison easier since there is no randomness in the selection of train/test splits.

[4]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/home/wiki.cgi?page=GENIA+Project

the named entity and two words on each side, suffix information, and positional information. We assume a two-phase model where boundary identification has already been performed, as in (Lee et al., 2004).

SVMs deliver high performance for the datasets we use so we employ SVMs as our base learner in the bulk of our experiments (maximum entropy models are used in Subsection 4.3). For selection of points to query, we use the approach that was used in (Tong and Koller, 2002; Schohn and Cohn, 2000; Campbell et al., 2000) of selecting the points that are closest to the current hyperplane. We use SVM^{light} (Joachims, 1999) for training the SVMs. For the smaller datasets (less than 50,000 examples in total), a batch size of 20 was used with an initial training set of size 100 and for the larger datasets (greater than 50,000 examples in total), a batch size of 200 was used with an initial training set of size 1000.

4.2 Main Results

Table 1 shows the results for all of our datasets. For each dataset, we report the average number of annotations[5] requested by each of the stopping methods as well as the average F-measure achieved by each of the stopping methods.[6]

There are two facts worth keeping in mind. First, the numbers in Table 1 are averages and therefore, sometimes two methods could have very similar average numbers of annotations but wildly different average F-measures (because one of the methods was consistently stopping around its average whereas the other was stopping way too early and way too late). Second, sometimes a method with a higher average number of annotations has a lower average F-measure than a method with a lower average number of annotations. This can be caused because of the first fact just mentioned about the numbers being averages and/or this can also be caused by the "less is more" phenomenon in active learning where often with less data, a higher-performing model is learned than with all the data; this was first reported in (Schohn and Cohn, 2000) and subsequently observed by many others (e.g., (Vlachos, 2008; Laws and Schütze, 2008)).

There are a few observations to highlight regarding the performance of the various stopping methods:

- SP is the most parsimonious method in terms of annotations. It stops the earliest and remarkably it is able to do so largely without sacrificing F-measure.
- All the methods except for SP and SC2000 are unstable in the sense that on at least one dataset they have a major failure, either stopping way too late and wasting large numbers of annotations (e.g. ZWH2008 and V2008 on TREC Spam) or stopping way too early and losing large amounts of F-measure (e.g. LS2008 on NER-Protein) .
- It's not always clear how to evaluate stopping methods because the tradeoff between the value of extra F-measure versus saving annotations is not clearly known and will be different for different applications and users.

This last point deserves some more discussion. In some cases it is clear that one stopping method is the best. For example, on WKB-Project, the SP method saves the most annotations *and* has the highest F-measure. But which method performs the best on NER-DNA? Arguments can reasonably be made for SP, SC2000, or ZWH2008 being the best in this case depending on what exactly the annotation/performance tradeoff is. A promising direction for research on AL stopping methods is to develop user-adjustable stopping methods that stop as aggressively as the user's annotation/performance preferences dictate.

One avenue of providing user-adjustable stopping is that if some methods are known to perform consistently in an aggressive manner against annotating

[5] Better evaluation metrics would use more refined measures of annotation effort than the number of annotations because not all annotations require the same amount of effort to annotate but lacking such a refined model for our datasets, we use number of annotations in these experiments.

[6] Tests of statistical significance are performed using matched pairs t tests at a 95% confidence level.

[7] (Vlachos, 2008) suggests using three drops in a row to detect a consistent drop in confidence so we do the same in our implementation of the method from (Vlachos, 2008).

[8] Following (Zhu et al., 2008b), we set the starting accuracy threshold to 0.9 when reimplementing their method.

[9] (Laws and Schütze, 2008) uses a window of size 100 and a threshold of 0.00005 so we do the same in our reimplementation of their method.

Task-Dataset	SP	V2008[7]	SC2000	ZWH2008[8]	LS2008[9]	All
TREC-SPAM	2100	**56000**	**3900**	**29220**	**3160**	56000
(10-fold AVG)	98.33	98.47	98.41	98.44	96.63	98.47
20Newsgroups	678	**181**	**1984**	1340	**1669**	11280
(20-cat AVG)	60.85	**18.06**	**55.43**	60.72	**54.79**	54.81
Spamassassin	326	**4362**	**862**	**398**	**1176**	5400
(10-fold AVG)	94.57	95.00	95.53	95.94	95.62	95.63
NER-protein	8720	**67220**	**17680**	**18580**	**2360**	67220
(10-fold AVG)	89.48	**90.28**	**90.38**	**90.31**	**76.47**	90.28
NER-DNA	4020	**67220**	**10640**	**7200**	**3900**	67220
(10-fold AVG)	82.40	**84.31**	**84.73**	**84.51**	**74.74**	84.31
NER-cellType	3840	**29600**	**5540**	11580	4580	67220
(10-fold AVG)	86.15	86.87	**87.19**	**87.32**	85.65	87.83
Reuters	484	**6762**	**1196**	**650**	**1272**	9580
(10-cat AVG)	74.29	**65.81**	**73.88**	76.77	74.00	75.64
WKB-Course	790	**184**	**1752**	**912**	**1740**	7420
(10-fold AVG)	83.12	**30.34**	**80.47**	83.16	**80.55**	80.19
WKB-Faculty	808	892	**1932**	**1062**	**1818**	7420
(10-fold AVG)	81.53	**40.14**	81.79	81.64	81.99	82.36
WKB-Project	646	916	**1358**	**794**	**1482**	7420
(10-fold AVG)	63.30	**25.33**	**58.11**	61.82	**59.30**	61.19
WKB-Student	1258	894	**2400**	**1468**	**2150**	7420
(10-fold AVG)	84.70	**50.66**	**83.46**	84.39	**83.19**	83.30
Average	2152	**21294**	**4477**	**6655**	2301	28509
(macro-avg)	81.70	**62.30**	80.85	82.27	**78.45**	81.27

Table 1: Methods for stopping AL. For each dataset, the average number of annotations at the automatically determined stopping points and the average F-measure at the automatically determined stopping points are displayed. **Bold entries** are statistically significantly different than SP (and non-bold entries are not). The Average row is simply an unweighted macro-average over all the datasets. The final column (labeled "All") represents standard fully supervised passive learning with the entire set of training data.

too much while others are known to perform consistently in a conservative manner, then users can pick the stopping criterion that's more suitable for their particular annotation/performance valuation. For this purpose, SP fills a gap as the other stopping criteria seem to be conservative in the sense defined in Section 1. SP, on the other hand, is more of an aggressive stopping criterion and is less likely to annotate data that is not needed.

A second avenue for providing user-adjustable stopping is a single stopping method that is itself adjustable. To this end, Section 4.3 shows how *intensity* and *longevity* provide levers that can be used to control the behavior of SP in a controlled fashion.

Sometimes viewing the stopping points of the various criteria on a graph with the active learning curve can help one visualize how the methods perform. Figure 2 shows the graph for a representative fold.[10] The x-axis measures the number of human annotations that have been requested so far. The y-axis measures performance in terms of F-Measure. The vertical lines are where the various stopping methods would have stopped AL if we hadn't continued the simulation. The figure reinforces and illustrates what we have seen in Table 1, namely that SP stops more aggressively than existing criteria and is able

[10]It doesn't make sense to show a graph for the average over cross validation because the average number of annotations at the stopping point may cross the learning curve at a completely misleading point. Consider a method that stops way too early and way too late at times.

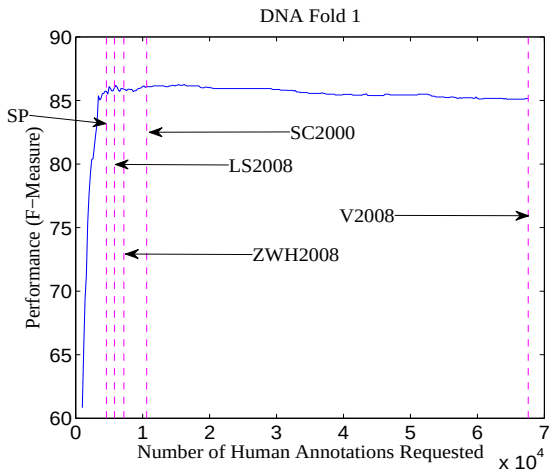


Figure 2: Graphic with stopping criteria in action for fold 1 of NER of DNA from the GENIA corpus. The x-axis ranges from 0 to 70,000.

to do so without sacrificing performance.

4.3 Additional Experiments

All of the additional experiments in this subsection were conducted on our least computationally demanding dataset, Spamassassin. The results in Tables 2 and 3 show how varying the intensity cutoff and the longevity requirement, respectively, of SP enable a user to control stopping behavior. Both methods enable a user to adjust stopping in a controlled fashion (without radical changes in behavior). Areas of future work include: combining the intensity and longevity methods for controlling behavior; and developing precise expectations on the change in behavior corresponding to changes in the intensity and longevity settings.

The results in Table 4 show results for different stop set sizes. Even with random selection of a stop set as small as 500, SP's performance holds fairly steady. This plus the fact that random selection of stop sets of size 2000 worked across all the folds of all the datasets in Table 1 show that in practice perhaps the simple heuristic of choosing a fairly large random set of points works well. Nonetheless, we think the size necessary will depend on the dataset and other factors such as the feature representation so more principled methods of determining the size and/or the makeup of the stop set are an area for future work. For example, construction techniques

Intensity	Annotations	F-Measure
K=99.5	364	96.01
K=99.0	326	94.57
K=98.5	304	95.59
K=98.0	262	93.75
K=97.5	242	93.35
K=97.0	224	90.91

Table 2: Controlling the behavior of stopping through the use of *intensity*. For Kappa intensity levels in $\{97.0, 97.5, 98.0, 98.5, 99.0, 99.5\}$, the 10-fold average number of annotations at the automatically determined stopping points and the 10-fold average F-measure at the automatically determined stopping points are displayed for the Spamassassin dataset.

Longevity	Annotations	F-Measure
k=1	284	95.17
k=2	318	94.95
k=3	326	94.57
k=4	336	95.40
k=5	346	96.41
k=6	366	94.53

Table 3: Controlling the behavior of stopping through the use of *longevity*. For window length k longevity levels in $\{1, 2, 3, 4, 5, 6\}$, the 10-fold average number of annotations at the automatically determined stopping points and the 10-fold average F-measure at the automatically determined stopping points are displayed for the Spamassassin dataset.

could be developed to create stop sets with high representativeness (in terms of feature space) density (meaning representativeness of stop set divided by size of stop set). For example, a possibility is to cluster examples before AL begins and then make sure the stop set contains examples from each of the clusters. Another possibility is to use a greedy algorithm where the stop set is iteratively grown where on each iteration the center of mass of the stop set in feature space is computed and an example in the unlabeled pool that is maximally far in feature space from this center of mass is selected for inclusion in the stop set. This could be useful for efficiency (in terms of getting the same stopping performance with a smaller stop set as could be achieved with a larger stop set) and also as a way to ensure adequate representation of the task space. The latter can be accom-

Task-Dataset	SP	V2008	ZWH2008	LS2008	All
Spamassassin	286	1208	**386**	**756**	5400
(10-fold AVG)	94.92	**89.89**	95.31	96.40	91.74

Table 5: Methods for stopping AL with maximum entropy as the base learner. For each stopping method, the average number of annotations at the automatically determined stopping point and the average F-measure at the automatically determined stopping point are displayed. **Bold entries** are statistically significantly different than SP (and non-bold entries are not). SC2000, the margin exhaustion method, is not shown since it can't be used with a non-margin-based learner. The final column (labeled "All") represents standard fully supervised passive learning with the entire set of training data.

Stop Set Size	Annotations	F-Measure
2500	326	95.58
2000	326	94.57
1500	314	95.00
1000	328	95.73
500	314	94.57

Table 4: Investigating the sensitivity to stop set size. For stop set sizes in {2500, 2000, 1500, 1000, 500}, the 10-fold average number of annotations at the automatically determined stopping points and the 10-fold average F-measure at the automatically determined stopping points are displayed for the Spamassassin dataset.

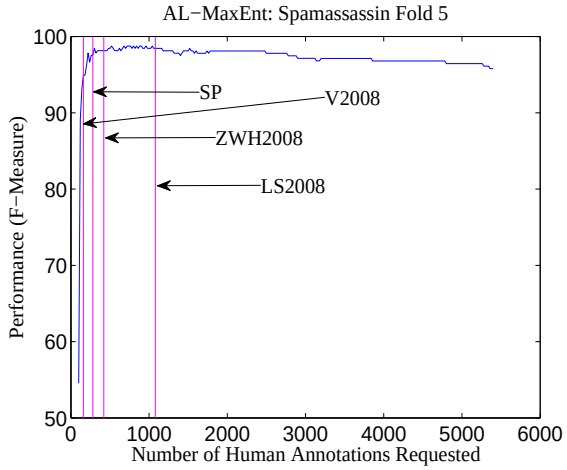


Figure 3: Graphic with stopping criteria in action for fold 5 of TC of the spamassassin corpus. The x-axis ranges from 0 to 6,000.

plished by perhaps continuing to add examples to the stop set until adding new examples is no longer increasing the representativeness of the stop set.

As one of the advantages of SP is that it's widely applicable, Table 5 shows the results when using maximum entropy models as the base learner during AL (the query points selected are those which the model is most uncertain about). The results reinforce our conclusions from the SVM experiments, with SP performing aggressively and all statistically significant differences in performance being in SP's favor. Figure 3 shows the graph for a representative fold.

5 Conclusions

Effective methods for stopping AL are crucial for realizing the potential annotation savings enabled by AL. A survey of existing stopping methods identified three areas where improvements are called for. The new stopping method based on Stabilizing Predictions (SP) addresses all three areas: SP is widely applicable, stable, and aggressive in saving annotations.

The empirical evaluation of SP and the existing methods was informative for evaluating the criteria but it was also informative for demonstrating the difficulties for rigorous objective evaluation of stopping criteria due to different annotation/performance tradeoff valuations. This opens up a future area for work on user-adjustable stopping. Two potential avenues for enabling user-adjustable stopping are a single criterion that is itself adjustable or a suite of methods with consistent differing levels of aggressiveness/conservativeness from which users can pick the one(s) that suit their annotation/performance tradeoff valuation. SP substantially widens the range of behaviors of existing methods that users can choose from. Also, SP's behavior itself can be adjusted through user-controllable parameters.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Jason Baldridge and Miles Osborne. 2008. Active learning and logarithmic opinion pools for hpsg parse selection. *Nat. Lang. Eng.*, 14(2):191–222.

Michael Bloodgood and K. Vijay-Shanker. 2009. Taking into account the differences between actively and passively acquired data: The case of active learning with support vector machines for imbalanced datasets. In *NAACL*.

Colin Campbell, Nello Cristianini, and Alex J. Smola. 2000. Query learning with large margin classifiers. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 111–118, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

J. Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46.

Gordon Cormack and Thomas Lynam. 2005. Trec 2005 spam track overview. In *TREC-14*.

Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. 1998. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, New York, NY, USA. ACM.

Seyda Ertekin, Jian Huang, Léon Bottou, and C. Lee Giles. 2007. Learning on the border: active learning in imbalanced data classification. In Mário J. Silva, Alberto H. F. Laender, Ricardo A. Baeza-Yates, Deborah L. McGuinness, Bjørn Olstad, Øystein Haug Olsen, and André O. Falcão, editors, *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, pages 127–136. ACM.

Thorsten Joachims. 1998. Text categorization with suport vector machines: Learning with many relevant features. In *ECML*, pages 137–142.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods – Support Vector Learning*, pages 169–184.

Florian Laws and Hinrich Schütze. 2008. Stopping criteria for active learning of named entity recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 465–472, Manchester, UK, August. Coling 2008 Organizing Committee.

Ki-Joong Lee, Young-Sook Hwang, Seonho Kim, and Hae-Chang Rim. 2004. Biomedical named entity recognition using two-phase model based on svms. *Journal of Biomedical Informatics*, 37(6):436–447.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. In *Proceedings of AAAI-98, Workshop on Learning for Text Categorization*.

Greg Schohn and David Cohn. 2000. Less is more: Active learning with support vector machines. In *Proc. 17th International Conf. on Machine Learning*, pages 839–846. Morgan Kaufmann, San Francisco, CA.

D. Sculley. 2007. Online active learning methods for fast label-efficient spam filtering. In *Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, USA.

Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research (JMLR)*, 2:45–66.

Andreas Vlachos. 2008. A stopping criterion for active learning. *Computer Speech and Language*, 22(3):295–312.

Jingbo Zhu and Eduard Hovy. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 783–790.

Jingbo Zhu, Huizhen Wang, and Eduard Hovy. 2008a. Learning a stopping criterion for active learning for word sense disambiguation and text classification. In *IJCNLP*.

Jingbo Zhu, Huizhen Wang, and Eduard Hovy. 2008b. Multi-criteria-based strategy to stop active learning for data annotation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1129–1136, Manchester, UK, August. Coling 2008 Organizing Committee.

Superior and Efficient Fully Unsupervised Pattern-based Concept Acquisition Using an Unsupervised Parser

Dmitry Davidov[1] **Roi Reichart**[1] **Ari Rappoport**[2]

[1]ICNC , [2]Institute of Computer Science
Hebrew University of Jerusalem
{dmitry@alice.nc|roiri@cs|arir@cs}.huji.ac.il

Abstract

Sets of lexical items sharing a significant aspect of their meaning (*concepts*) are fundamental for linguistics and NLP. Unsupervised concept acquisition algorithms have been shown to produce good results, and are preferable over manual preparation of concept resources, which is labor intensive, error prone and somewhat arbitrary. Some existing concept mining methods utilize supervised language-specific modules such as POS taggers and computationally intensive parsers.

In this paper we present an efficient fully unsupervised concept acquisition algorithm that uses syntactic information obtained from a fully unsupervised parser. Our algorithm incorporates the bracketings induced by the parser into the meta-patterns used by a symmetric patterns and graph-based concept discovery algorithm. We evaluate our algorithm on very large corpora in English and Russian, using both human judgments and WordNet-based evaluation. Using similar settings as the leading fully unsupervised previous work, we show a significant improvement in concept quality and in the extraction of multiword expressions. Our method is the first to use fully unsupervised parsing for unsupervised concept discovery, and requires no language-specific tools or pattern/word seeds.

1 Introduction

Comprehensive lexical resources for many domains and languages are essential for most NLP applications. One of the most utilized types of such resources is a repository of *concepts*: sets of lexical items sharing a significant aspect of their meanings (e.g., types of food, tool names, etc).

While handcrafted concept databases (e.g., WordNet) are extensively used in NLP, manual compilation of such databases is labor intensive, error prone, and somewhat arbitrary. Hence, for many languages and domains great efforts have been made for automated construction of such databases from available corpora. While language-specific and domain-specific studies show significant success in development of concept discovery frameworks, the majority of domains and languages remain untreated. Hence there is a need for a framework that performs well for many diverse settings and is as unsupervised and language-independent as possible.

Numerous methods have been proposed for seed-based concept extraction where a set of concept patterns (or rules), or a small set of seed words for each concept, is provided as input to the concept acquisition system. However, even simple definitions for concepts are not always available.

To avoid requiring this type of input, a number of distributional and pattern-based methods have been proposed for fully unsupervised seed-less acquisition of concepts from text. Pattern-based algorithms were shown to obtain high quality results while being highly efficient in comparison to distributional methods. Such fully unsupervised methods do not incorporate any language-specific parsers or taggers, so can be successfully applied to diverse languages.

However, unsupervised pattern-based methods suffer from several weaknesses. Thus they are frequently restricted to single-word terms and are unable to discover multiword expressions in efficient and precise manner. They also usually ignore potentially useful part-of-speech and other syntactic information. In order to address these weaknesses, several studies utilize language-specific parsing or

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 48–56,
Boulder, Colorado, June 2009.

tagging systems in concept acquisition. Unfortunately, while improving results, this heavily affects the language- and domain- independence of such frameworks, and severely impacts efficiency since even shallow parsing is computationally demanding.

In this paper we present a method to utilize the information induced by unsupervised parsers in an unsupervised pattern-based concept discovery framework. With the recent development of fast fully unsupervised parsers, it is now possible to add parser-based information to lexical patterns while keeping the language-independence of the whole framework and still avoiding heavy computational costs. Specifically, we incorporate the bracketings induced by the parser into the meta-patterns used by a symmetric patterns and graph-based unsupervised concept discovery algorithm.

We performed a thorough evaluation on two English corpora (the BNC and a 68GB web corpus) and on a 33GB Russian corpus. Evaluations were done using both human judgments and WordNet, in similar settings as that of the leading unsupervised previous work. Our results show that utilization of unsupervised parser both improves the assignment of single-word terms to concepts and allows high-precision discovery and assignment of of multiword expressions to concepts.

2 Previous Work

Much work has been done on lexical acquisition of all sorts and the acquisition of concepts in particular. Concept acquisition methods differ in the type of corpus annotation and other human input used, and in their basic algorithmic approach. Some methods directly aim at concept acquisition, while the direct goal in some is the construction of hyponym ('is-a') hierarchies. A subtree in such a hierarchy can be viewed as defining a concept.

A major algorithmic approach is to represent word contexts as vectors in some space and use distributional measures and clustering in that space. Pereira (1993), Curran (2002) and Lin (1998) use syntactic features in the vector definition. (Pantel and Lin, 2002) improves on the latter by clustering by committee. Caraballo (1999) uses conjunction and appositive annotations in the vector representation. Several studies avoid requiring any syntactic annotation. Some methods are based on decomposition of a lexically-defined matrix (by SVD, PCA etc), e.g. (Schütze, 1998; Deerwester et al., 1990).

While great effort has been made for improving the computational complexity of distributional methods (Gorman and Curran, 2006), they still remain highly computationally intensive in comparison to pattern approaches (see below), and most of them do not scale well for very large datasets.

The second main approach is to use lexico-syntactic patterns. Patterns have been shown to produce more accurate results than feature vectors, at a lower computational cost on large corpora (Pantel et al., 2004). Since (Hearst, 1992), who used a manually prepared set of initial lexical patterns, numerous pattern-based methods have been proposed for the discovery of concepts from seeds. Other studies develop concept acquisition for on-demand tasks where concepts are defined by user-provided seeds. Many of these studies utilize information obtained by language-specific parsing and named entity recognition tools (Dorow et al., 2005). Pantel et al. (2004) reduce the depth of linguistic data used, but their method requires POS tagging.

TextRunner (Banko et al., 2007) utilizes a set of pattern-based seed-less strategies in order to extract relational tuples from text. However, this system contains many language-specific modules, including the utilization of a parser in one of the processing stages. Thus the majority of the existing pattern-based concept acquisition systems rely on pattern/word seeds or supervised language-specific tools, some of which are very inefficient.

Davidov and Rappoport (2006) developed a framework which discovers concepts based on high frequency words and symmetry-based pattern graph properties. This framework allows a fully unsupervised seed-less discovery of concepts without relying on language-specific tools. However, it completely ignores potentially useful syntactic or morphological information.

For example, the pattern 'X and his Y' is useful for acquiring the concept of family member types, as in "his siblings and his parents'. Without syntactic information, it can capture noise, as in "... in ireland) and his wife)" (parentheses denote syntactic constituent boundaries). As another example, the useful symmetric pattern "either X or Y" can appear in both good examples ("choose either *Chihuahua*

or *Collie*.") and bad ones ("either *Collie* or *Australian* Bulldog"). In the latter case, the algorithm both captures noise ("Australlian" is now considered as a candidate for the 'dog type' concept), and misses the discovery of a valid multiword candidate ("Australlian Bulldog"). While symmetry-based filtering greatly reduces such noise, the basic problem remains. As a result, incorporating at least some parsing information in a language-independent and efficient manner could be beneficial.

Unsupervised parsing has been explored for several decades (see (Clark, 2001; Klein, 2005) for recent reviews). Recently, unsupervised parsers have for the first time outperformed the right branching heuristic baseline for English. These include CCM (Klein and Manning, 2002), the DMV and DMV+CCM models (Klein and Manning, 2004), (U)DOP based models (Bod, 2006a; Bod, 2006b; Bod, 2007), an exemplar based approach (Dennis, 2005), guiding EM using contrastive estimation (Smith and Eisner, 2006), and the incremental parser of Seginer (2007) which we use here. These works learn an unlabeled syntactic structure, dependency or constituency. In this work we use constituency trees as our syntactic representation.

Another important factor in concept acquisition is the source of textual data used. To take advantage of the rapidly expanding web, many of the proposed frameworks utilize web queries rather than local corpora (Etzioni et al., 2005; Davidov et al., 2007; Pasca and Van Durme, 2008; Davidov and Rappoport, 2009). While these methods have a definite practical advantage of dealing with the most recent and comprehensive data, web-based evaluation has some methodological drawbacks such as limited repeatability (Kilgarriff, 2007). In this study we apply our framework on offline corpora in settings similar to that of previous work, in order to be able to make proper comparisons.

3 Efficient Unsupervised Parsing

Our method utilizes the information induced by unsupervised parsers. Specifically, we make use of the bracketings induced by Seginer's parser[1] (Seginer, 2007). This parser has advantages in three major aspects relevant to this paper.

First, it achieves state of the art unsupervised parsing performance: its F-score[2] is 75.9% for sentences of up to 10 words from the PennTreebank Wall Street Journal corpus (WSJ) (Marcus, 1993), and 59% for sentences of the same length from the German NEGRA (Brants, 1997) corpus. These corpora consists of newspaper texts.

Second, to obtain good results, manually created POS tags are used as input in all the unsupervised parsers mentioned above except of Seginer's, which uses raw sentences as input. (Headden et al., 2008) have shown that the performance of algorithms that require POS tags substantially decreases when using POS tags induced by unsupervised POS taggers instead of manually created ones. Seginer's incremental parser is therefore the only *fully* unsupervised parser providing high quality parses.

Third, Seginer's parser is extremely fast. During its initial stage, the parser builds a lexicon. Our Pentium 2.8GHB machines with 4GHB RAM can store in memory the lexicon created by up to 0.2M sentences. We thus divided our corpora to batches of 0.2M sentences and parsed each of them separately. Note that in this setup parsing quality might be even better than the quality reported in (Seginer, 2007), since in the setup reported in that paper the parser was applied to a few thousand sentences only. On average, the parsing time of a single batch was 5 minutes (run time did not significantly differ across batches and corpora).

Parser description. The parser utilizes the novel common-cover link representation for syntactic structure. This representation resembles dependency structure but unlike the latter, it can be translated into a constituency tree, which is the syntactic representation we use in this work.

The parsing algorithm creates the common-cover links structure of a sentence in an incremental manner. This means that the parser reads the words of a sentence one after the other and, as each word is read, it is only allowed to add links that have one of their ends at that words (and update existing ones). Words which have not yet been read are not avail-

[1] The parser is freely available at *http://staff.science.uva.nl/∼yseginer/ccl*

[2] $F = \frac{2 \cdot R \cdot P}{R+P}$, where R and P are the recall and precision of the parsers' bracketing compared to manually created bracketing of the same text. This is the accepted measure for parsing performance (see (Klein, 2005)).

able to the parser at this stage. This restriction is inspired by psycholinguistics research which suggests that humans process language incrementally. This results in a significant restriction of the parser's search space, which is the reason it is so fast.

During its initial stage the parser builds a lexicon containing, for each word, statistics helping the decision of whether to link that word to other words. The lexicon is updated as any new sentence is read. Lexicon updating is also done in an incremental manner so this stage is also very fast.

4 Unsupervised Pattern Discovery

In the first stage of our algorithm, we run the unsupervised parser on the corpus in order to produce a bracketing structure for each sentence. In the second stage, described here, we use these bracketings in order to discover, in a fully unsupervised manner, patterns that could be useful for concept mining.

Our algorithm is based on the concept acquisition method of (Davidov and Rappoport, 2006). We discover patterns that connect terms belonging to the same concept in two main stages: discovery of pattern candidates, and identification of the symmetric patterns among the candidates.

Pattern candidates. A major idea of (Davidov and Rappoport, 2006) is that a few dozen high frequency words (HFW) such as 'and' and 'is' connect other, less frequent content terms into relationships. They define *meta-patterns*, which are short sequences of H's and C's, where H is a slot for a HFW and C is a slot for a content word (later to become a word belonging to a discovered concept). Their method was shown to produce good results. However, the fact that it does not consider any syntactic information causes problems. Specifically, it does not consider the constituent structure of the sentence. Meta-patterns that cross constituent boundaries are likely to generate noise – two content words (C's) in a meta-pattern that belong to different constituents are likely to belong to different concepts as well. In addition, meta-patterns that do not occupy a full constituent are likely to 'cut' multiword expressions (MWEs) into two parts, one part that gets treated as a valid C word and one part that is completely ignored.

The main idea in the present paper is to use the bracketings induced by unsupervised parsers in order to avoid the problems above. We utilize bracketing boundaries in our meta-patterns in addition to HFW and C slots. In other words, their original meta-patterns are totally lexical, while ours are lexico-syntactic meta-patterns. We preserve the attractive properties of meta-patterns, because both HFWs and bracketings can be found or computed in a language independent manner and very efficiently.

Concretely, we define a HFW as a word appearing more than T_H times per million words, and a C as a word or multiword expression containing up to 4 words, appearing less than T_C times per million.

We require that our patterns include two slots for C's, separated by at least a single HFW or bracket. We allow separation by a single bracket because the lowest level in the induced bracketing structure usually corresponds to lexical items, while higher levels correspond to actual syntactic constituents.

In order to avoid truncation of multiword expressions, we also require the meta pattern to start and end by a HFW or bracket. Thus our meta-patterns match the following regular expression:

$$\{H|B\}^* \; C_1 \; \{H|B\}^+ \; C_2 \; \{H|B\}^*$$

where "*" means zero or more times, and "+" means one or more time and B can be "(","(" brackets produced by the parser (in these patterns we do not need to guarantee that brackets match properly). Examples of such patterns include "$((C_1)in\ C_2))$", "$(C_1)(such(as(((C_2)$", and "$(C_1)and(C_2)$"[3]. We dismiss rare patterns that appear less than T_P times per million words.

Symmetric patterns. Many of the pattern candidates discovered in the previous stage are not usable. In order to find a usable subset, we focus on the symmetric patterns. We define a symmetric pattern as a pattern in which the same pair of terms (C words) is likely to appear in both left-to-right and right-to-left orders. In order to identify symmetric patterns, for each pattern we define a pattern graph $G(P)$, as proposed by (Widdows and Dorow, 2002). If term pair (C_1, C_2) appears in pattern P in some context,

[3]This paper does not use any punctuation since the parser is provided with sentences having all non-alphabetic characters removed. We assume word separation. $C_{1,2}$ can be a word or a multiword expression.

we add nodes c_1, c_2 to the graph and a directed edge $E_P(c_1, c_2)$ between them. In order to select symmetric patterns, we create such a pattern graph for every discovered pattern, and create a symmetric subgraph SymG(P) in which we take only bidirectional edges from $G(P)$. Then we compute three measures for each pattern candidate as proposed by (Davidov and Rappoport, 2006):

$$M_1(P) := \frac{|\{c_1|\exists c_2 E_P(c_1, c_2) \wedge \exists c_3 E_P(c_3, c_1)\}|}{|Nodes(G(P))|}$$

$$M_2(P) := \frac{|Nodes(SymG(P))|}{|Nodes(G(P))|}$$

$$M_3(P) := \frac{|Edges(SymG(P))|}{|Edges(G(P))|}$$

For each measure, we prepare a sorted list of all candidate patterns. We remove patterns that are not in the top Z_T (we use 100, see Section 6) in any of the three lists, and patterns that are in the bottom Z_B in at least one of the lists.

5 Concept Discovery

At the end of the previous stage we have a set of symmetric patterns. We now use them in order to discover concepts. The concept discovery algorithm is essentially the same as used by (Davidov and Rappoport, 2006) and has some similarity with the one used by (Widdows and Dorow, 2002). In this section we outline the algorithm.

The clique-set method. The utilized approach to concept discovery is based on connectivity structures in the all-pattern term relationship graph G, resulting from merging all of the single-pattern graphs for symmetric patterns selected in the previous stage. The main observation regarding G is that highly interconnected words are good candidates to form a concept. We find all strong n-cliques (subgraphs containing n nodes that are all interconnected in both directions). A clique Q defines a concept that contains all of the nodes in Q plus all of the nodes that are (1) at least unidirectionally connected to all nodes in Q, and (2) bidirectionally connected to at least one node in Q. Using this definition, we create a concept for each such clique.

Note that a single term can be assigned to several concepts. Thus a clique based on a connection of the word 'Sun' to 'Microsoft' can lead to a concept of computer companies, while the connection of 'Sun' to 'Earth' can lead to a concept of celestial bodies.

Reducing noise: merging and windowing. Since any given term can participate in many cliques, the algorithm creates overlapping categories, some of which redundant. In addition, due to the nature of language and the imperfection of the corpus some noise is obviously to be expected. We enhance the quality of the obtained concepts by merging them and by windowing on the corpus. We merge two concepts Q, R, iff there is more than a 50% overlap between them: $(|Q \bigcap R| > |Q|/2) \wedge (|Q \bigcap R| > |R|/2)$. In order to increase concept quality and remove concepts that are too context-specific, we use a simple corpus windowing technique. Instead of running the algorithm of this section on the whole corpus, we divide the corpus into windows of equal size and perform the concept discovery algorithm of this section (without pattern discovery) on each window independently. We now have a set of concepts for each window. For the final set, we select only those concepts that appear in at least two of the windows. This technique reduces noise at the potential cost of lowering coverage.

A decrease in the number of windows should produce more noisy results, while discovering more concepts and terms. In the next section we show that while windowing is clearly required for a large corpus, incorporation of parser data increases the quality of the extracted corpus to the point where windowing can be significantly reduced.

6 Results

In order to estimate the quality of concepts and to compare it to previous work, we have performed both automatic and human evaluation. Our basic comparison was to (Davidov and Rappoport, 2006) (we have obtained their data and utilized their algorithm), where we can estimate if incorporation of parser data can solve some fundamental weaknesses of their framework. In the following description, we call their algorithm **P** and our parser-based framework **P+**. We have also performed an indirect comparison to (Widdows and Dorow, 2002).

While there is a significant number of other related studies[4] on concept acquisition (see Section 2),

[4]Most are supervised and/or use language-specific tools.

direct or even indirect comparison to these works is problematic due to difference in corpora, problem definitions and evaluation strategies. Below we describe the corpora and parameters used in our evaluation and then show and discuss WordNet-based and Human evaluation settings and results.

Corpora. We performed in-depth evaluation in two languages, English and Russian, using three corpora, two for English and one for Russian. The first English corpus is the BNC, containing about 100M words. The second English corpus, DMOZ(Gabrilovich and Markovitch, 2005), is a web corpus obtained by crawling URLs in the Open Directory Project (dmoz.org), resulting in 68GB containing about 8.2G words from 50M web pages. The Russian corpus (Davidov and Rappoport, 2006) was assembled from web-based Russian repositories, to yield 33GB and 4G words. All of these corpora were also used by (Davidov and Rappoport, 2006) and BNC was used in similar settings by (Widdows and Dorow, 2002).

Algorithm parameters. The thresholds T_H, T_C, T_P, Z_T, Z_B, were determined mostly by practical memory size considerations: we computed thresholds that would give us the maximal number of terms, while enabling the pattern access table to reside in main memory. The resulting numbers are $100, 50, 20, 100, 100$. Corpus window size was determined by starting from a small window size, extracting at random a single window, running the algorithm, and iterating this process with increased $\times 2$ window sizes until reaching a desired vocabulary concept participation percentage (before windowing) (i.e., x% of the different words in the corpus participate in terms assigned into concepts. We used 5%.). We also ran the algorithm without windowing in order to check how well the provided parsing information can help reduce noise. Among the patterns discovered are the ubiquitous ones containing "and","or", e.g. '((X) or (a Y))', and additional ones such as 'from (X) to (Y)'.

Influence of parsing data on number of discovered concepts. Table 1 compares the concept acquisition framework with (P+) and without (P) utilization of parsing data.

We can see that the amount of different words covered by discovered concepts raises nearly 1.5-fold when we utilize patterns based on parsing data in comparison to pure HFW patterns used in previous work. We can also see nearly the same increase in average concept size. At the same time we observe about 15% reduction in the total number of discovered concepts.

	V	W		C		AS	
		P	P+	P	P+	P	P+
DMOZ	16	330	**504**	**142**	130	12.8	**16.0**
BNC	0.3	25	**42**	**9.6**	8.9	10.2	**15.6**
Russ.	10	235	**406**	**115**	96	11.6	**15.1**

Table 1: Results for concept discovery with (P+) and without (P) utilization of parsing data. V is the total number (millions) of different words in the corpus. W is the number (thousands) of words belonging to at least one of the terms for one of the concepts. C is the number (thousands) of concepts (after merging and windowing). AS is the average(words) category size.

There are two opposite factors in P+ which may influence the number of concepts, their size and coverage in comparison to P. On one hand, utilization of more restricted patterns that include parsing information leads to a reduced number of concept term instances being discovered. Thus, the P+ pattern "(X (or (a Y))" will recognize "(TV (or (a movie))" instance and will miss "(lunch) or (a snack))", while the P pattern "X or a Y" will capture both. This leads to a decrease in the number of discovered concepts.

On the other hand, P+ patterns, unlike P ones, allow the extraction of multiword expressions[5], and indeed more than third of the discovered terms using P+ were MWEs. Utilization of MWEs not only allows to cover a greater amount of different words, but also increases the number of discovered concepts since new concepts can be found using cliques of newly discovered MWEs. From the results, we can see that for a given concept size and word coverage, the ability to discover MWEs overcomes the disadvantage of ignoring potentially useful concepts.

Human judgment evaluation. Our human judgement evaluation closely followed the protocol (Davidov and Rappoport, 2006).

We used 4 subjects for evaluation of the English

[5]While P method can potentially be used to extract MWEs, preliminary experimentation shows that without significant modification, quality of MWEs obtained by P is very low in comparison to P+

concepts and 4 subjects for Russian ones. In order to assess subjects' reliability, we also included random concepts (see below). The goal of the experiment was to examine the differences between the P+ and P concept acquisition frameworks. Subjects were given 50 triplets of words and were asked to rank them using the following scale: (1) the words definitely share a significant part of their meaning; (2) the words have a shared meaning but only in some context; (3) the words have a shared meaning only under a very unusual context/situation; (4) the words do not share any meaning; (5) I am not familiar enough with some/all of the words.

The 50 triplets were obtained as follows. We have randomly selected 40 concept pairs (C+,C): C+ in P+ and C in P using five following restrictions: (1) concepts should contain at least 10 words; (2) for a selected pair, C+ should share at least half of its single-word terms with C, and C should share at least half of its words with C+; (3) C+ should contain at least 3 MWEs; (4) C should contain at least 3 words not appearing in C+; (5) C+ should contain at least 3 single-word terms not appearing in C.

These restrictions allow to select concept pairs such that C+ is similar to C while they still carry enough differences which can be examined. We selected the triplets as following: for pairs (C+, C) ten triplets include terms appearing in both C+ and C (*Both* column in Table 2), ten triplets include single-word terms appearing in C+ but not C (*P+ single* column), ten triplets include single-word terms appearing in C but not C+ (*P* column), ten triplets include MWEs appearing in C+ (*P+ mwe* column) and ten triplets include random terms obtained from P+ concepts (*Rand* column).

	P+		P	Both	Rand
	mwe	single			
% shared meaning					
DMOZ	85	**88**	68	81	6
BNC	85	**90**	61	88	0
Russ.	89	**95**	70	93	11
triplet score (1-4)					
DMOZ	1.7	**1.4**	2.5	1.7	3.8
BNC	1.6	**1.3**	2.1	1.5	4.0
Russ.	1.5	**1.1**	2.0	1.3	3.7

Table 2: Results of evaluation by human judgment of three data sets. P+ single/mwe: single-word/MWE terms existing only in P+ concept; P: single-word terms existing only in P concept; Both: terms existing in both concepts; Rand: random terms. See text for detailed explanations.

The first part of Table 2 gives the average percentage of triplets that were given scores of 1 or 2 (that is, 'significant shared meaning'). The second part gives the average score of a triplet (1 is best). In these lines scores of 5 were not counted. Inter-evaluator Kappa between scores are 0.68/0.75/0.76 for DMOZ, BNC and Russian respectively. We can see that terms selected by P and skipped by P+ receive low scores, at the same time even single-word terms selected by P+ and skipped by P show very high scores. This shows that using parser data, the proposed framework can successfully avoid selection of erroneous terms, while discovering high-quality terms missed by P. We can also see that P+ performance on MWEs, while being slightly inferior to the one for single-word terms, still achieves results comparable to those of single-word terms.

Thus our algorithm can greatly improve the results not only by discovering of MWEs but also by improving the set of single word concept terms.

WordNet-based evaluation. The major guideline in this part of the evaluation was to compare our results with previous work (Davidov and Rappoport, 2006; Widdows and Dorow, 2002) without the possible bias of human evaluation. We have followed their methodology as best as we could, using the same WordNet (WN) categories and the same corpora. This also allows indirect comparison to several other studies, thus (Widdows and Dorow, 2002) reports results for an LSA-based clustering algorithm that are vastly inferior to the pattern-based ones.

The evaluation method is as follows. We took the exact 10 WN subsets referred to as 'subjects' in (Widdows and Dorow, 2002), and removed all multi-word items. We then selected at random 10 pairs of words from each subject. For each pair, we found the largest of our discovered concepts containing it. The various morphological forms or clear typos of the same word were treated as one in the evaluation.

We have improved the evaluation framework for Russian by using the Russian WordNet (Gelfenbeynand et al., 2003) instead of back-translations as done in (Davidov and Rappoport, 2006). Preliminary examination shows that this has no apparent effect on the results.

For each found concept C containing N words, we computed the following: (1) Precision: the num-

ber of words present in both C and WN divided by N; (2) Precision*: the number of correct words divided by N. Correct words are either words that appear in the WN subtree, or words whose entry in the American Heritage Dictionary or the Britannica directly defines them as belonging to the given class (e.g., 'murder' is defined as 'a crime'). This was done in order to overcome the relative poorness of WN; (3) Recall: the number of words present in both C and WN divided by the number of words in WN; (4) The percentage of correctly discovered words (according to Precision*) that are not in WN.

Table 3 compares the macro-average of these 10 categories to corresponding related work. We do not

	Prec.	Prec.*	Rec.	%New
DMOZ				
P	**79.8**	86.5	22.7	2.5
P+	79.5	**91.3**	**28.6**	**3.7**
BNC				
P	92.76	95.72	7.22	0.4
P+	**93.0**	**96.1**	**14.6**	**1.7**
Widdows	82.0	-	-	-
Russian				
P	82.39	89.64	20.03	2.1
P+	**83.5**	**92.6**	**29.6**	**4.0**

Table 3: WordNet evaluation in comparison to P (Davidov and Rappoport, 2006) and to Widdows(Widdows and Dorow, 2002). Columns show average precision, precision* (as defined in text), recall, and % of new words added to corresponding WN subtree.

observe apparent rise in precision when comparing P+ and P, but we can see significant improvement in both recall and precision* for all of three corpora. In combination with human judgement results, this suggests that the P+ framework successfully discovers more correct terms not present in WN. This causes precision to remain constant while precision* improves significantly. Rise in recall also shows that the P+ framework can discover significantly more correct terms from the same data.

Windowing requirement. As discussed in Section 5, windowing is required for successful noise reduction. However, due to the increase in pattern quality with parser data, it is likely that less noise will be captured by the discovered patterns. Hence, windowing could be relaxed allowing to obtain more data with sufficiently high precision.

In order to test this issue we applied our algorithms on the DMOZ corpus with 3 different windowing settings: (1) choosing window size as described above; (2) using $\times 4$ larger window; (3) avoiding windowing altogether. Each time we randomly sampled a set of 100 concepts and tagged (by the authors) noisy ones. A concept is considered to be noisy if it has at least 3 words unrelated to each other. Table 4 shows results of this test.

	Reg. Window	$\times 4$ Window	No windowing
P	4	18	33
P+	4	5	21

Table 4: Percentage of noisy concepts as a function of windowing.

We can see that while windowing is still essential even with available parser data, using this data we can significantly reduce windowing requirements, allowing us to discover more concepts from the same data.

Timing requirements are modest, considering we parsed such large amounts of data. BNC parsing took 45 minutes, and the total single-machine processing time for the 68Gb DMOZ corpus was 4 days[6]. In comparison, a state-of-art supervised parser (Charniak and Johnson, 2005) would process the same amount of data in 1.3 years[7].

7 Discussion

We have presented a framework which utilizes an efficient fully unsupervised parser for unsupervised pattern-based discovery of concepts. We showed that utilization of unsupervised parser in pattern acquisition not only allows successful extraction of MWEs but also improves the quality of obtained concepts, avoiding noise and adding new terms missed by the parse-less approach. At the same time, the framework remains fully unsupervised, allowing its straightforward application to different languages as supported by our bilingual evaluation.

This research presents one more step towards the merging of fully unsupervised techniques for lexical acquisition, allowing to extract semantic data without strong assumptions on domain or language. While we have aimed for concept acquisition, the proposed framework can be also useful for extraction of different types of lexical relationships, both among concepts and between concept terms.

[6] In fact, we used a PC cluster, and all 3 corpora were parsed in 15 hours.

[7] Considering the reported parsing rate of 10 sentences per second

References

Mishele Banko, Michael J Cafarella , Stephen Soderland, Matt Broadhead, Oren Etzioni, 2007. Open Information Extraction from the Web. *IJCAI '07*.

Rens Bod, 2006a. An All-Subtrees Approach to Unsupervised Parsing. *ACL '06*.

Rens Bod, 2006b. Unsupervised Parsing with U-DOP. *CoNLL X*.

Rens Bod, 2007. Is the End of Supervised Parsing in Sight? *ACL '07*.

Thorsten Brants, 1997. The NEGRA Export Format. *CLAUS Report, Saarland University*.

Sharon Caraballo, 1999. Automatic Construction of a Hypernym-labeled Noun Hierarchy from Text. *ACL '99*.

Eugene Charniak and Mark Johnson, 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. *ACL '05*.

Alexander Clark, 2001. *Unsupervised Language Acquisition: Theory and Practice*. Ph.D. thesis, University of Sussex.

James R. Curran, Marc Moens, 2002. Improvements in Automatic Thesaurus Extraction *SIGLEX 02'*, 59–66.

Dmitry Davidov, Ari Rappoport, 2006. Efficient Unsupervised Discovery of Word Categories using Symmetric Patterns and High Frequency Words. *COLING-ACL '06*.

Dmitry Davidov, Ari Rappoport, Moshe Koppel, 2007. Fully Unsupervised Discovery of Concept-Specific Relationships by Web Mining. *ACL '07*.

Dmitry Davidov, Ari Rappoport, 2009. Translation and Extension of Concepts Across Languages. *EACL '09*.

Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, Richard Harshman, 1990. Indexing by Latent Semantic Analysis. *J. of the American Society for Info. Science*, 41(6):391–407.

Simon Dennis, 2005. An exemplar-based approach to unsupervised parsing. *Proceedings of the 27th Conference of the Cognitive Science Society*.

Beate Dorow, Dominic Widdows, Katarina Ling, Jean-Pierre Eckmann, Danilo Sergi, Elisha Moses, 2005. Using curvature and Markov clustering in Graphs for Lexical Acquisition and Word Sense Discrimination. *MEANING '05*.

Oren Etzioni, Michael Cafarella, Doug Downey, S. Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel Weld, Alexander Yates, 2005. Unsupervised Named-entity Extraction from the Web: An Experimental Study. *Artificial Intelligence, 165(1):91134*.

Evgeniy Gabrilovich, Shaul Markovitch, 2005. Feature Generation for Text Categorization Using World Knowledge. *IJCAI '05*.

Ilya Gelfenbeyn, Artem Goncharuk, Vladislav Lehelt, Anton Lipatov, Victor Shilo, 2003. Automatic Translation of WordNet Semantic Network to Russian Language (in Russian) *International Dialog 2003 Workshop*.

James Gorman, James R. Curran, 2006. Scaling Distributional Similarity to Large Corpora. *COLING-ACL '06*.

William P. Headden III, David McClosky and Eugene Charniak, 2008. Evaluating Unsupervised Part-of-Speech tagging for Grammar Induction. *COLING '08*.

Marti Hearst, 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. *COLING '92*.

Adam Kilgarriff, 2007. Googleology is Bad Science. *Computational Linguistics '08, Vol.33 No. 1,pp147-151*. .

Dan Klein and Christopher Manning, 2002. A generative constituent-context model for improved grammar induction. *Proc. of the 40th Meeting of the ACL*.

Dan Klein and Christopher Manning, 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. *ACL '04*.

Dan Klein, 2005. *The unsupervised learning of natural language structure.* Ph.D. thesis, Stanford University.

Hang Li, Naoki Abe, 1996. Clustering Words with the MDL Principle. *COLING '96*.

Dekang Lin, 1998. Automatic Retrieval and Clustering of Similar Words. *COLING '98*.

Marcus Mitchell P., Beatrice Santorini and Mary Ann Marcinkiewicz, 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330

Marius Pasca, Benjamin Van Durme, 2008. Weakly-supervised Acquisition of Open-domain Classes and Class Attributes from Web Documents and Query Logs. *ACL 08*.

Patrick Pantel, Dekang Lin, 2002. Discovering Word Senses from Text. *SIGKDD '02*.

Patrick Pantel, Deepak Ravichandran, Eduard Hovy, 2004. Towards Terascale Knowledge Acquisition. *COLING '04*.

Fernando Pereira, Naftali Tishby, Lillian Lee, 1993. Distributional Clustering of English Words. *ACL '93*.

Hinrich Schütze, 1998. Automatic Word Sense Discrimination. *Computational Linguistics* , 24(1):97–123.

Yoav Seginer, 2007. Fast Unsupervised Incremental Parsing. *ACL '07*.

Noah A. Smith and Jason Eisner, 2006. Annealing Structural Bias in Multilingual Weighted Grammar Induction . *ACL '06*.

Dominic Widdows, Beate Dorow, 2002. A Graph Model for Unsupervised Lexical Acquisition. *COLING '02*.

Representing words as regions in vector space

Katrin Erk
University of Texas at Austin
katrin.erk@mail.utexas.edu

Abstract

Vector space models of word meaning typically represent the meaning of a word as a vector computed by summing over all its corpus occurrences. Words close to this point in space can be assumed to be similar to it in meaning. But how far around this point does the region of similar meaning extend? In this paper we discuss two models that represent word meaning as *regions* in vector space. Both representations can be computed from traditional point representations in vector space. We find that both models perform at over 95% F-score on a token classification task.

1 Introduction

Vector space models of word meaning (Lund and Burgess, 1996; Landauer and Dumais, 1997; Lowe, 2001; Jones and Mewhort, 2007; Sahlgren and Karlgren, 2005) represent words as points in a high-dimensional semantic space. The dimensions of the space represent the contexts in which each target word has been observed. Distance between vectors in semantic space predicts the degree of semantic similarity between the corresponding words, as words with similar meaning tend to occur in similar contexts. Because of this property, vector space models have been used successfully both in computational linguistics (Manning et al., 2008; Snow et al., 2006; Gorman and Curran, 2006; Schütze, 1998) and in cognitive science (Landauer and Dumais, 1997; Lowe and McDonald, 2000; McDonald and Ramscar, 2001). Given the known problems with defining globally appropriate senses (Kilgarriff, 1997; Hanks, 2000), vector space models are especially interesting for their ability to represent word meaning without relying on dictionary senses.

Vector space models typically compute one vector per target word (what we will call *word type* vectors), summing co-occurrence counts over all corpus tokens of the target. If the target word is polysemous, the representation will constitute a union over the uses or senses of the word. Such a model does not provide information on the amount of *variance in each dimension*: Do values on each dimension vary a lot across occurrences of the target? Also, it does not provide information on *co-occurrences of feature values* in occurrences of the target. To encode these two types of information, we study richer models of word meaning in vector space beyond single point representations.

Many models of categorization in psychology represent a concept as a region, characterized by feature vectors with dimension weights (Smith et al., 1988; Hampton, 1991; Nosofsky, 1986). Taking our cue from these approaches, we study two models that represent a word as a *region* in vector space rather than a point. The first model is one that we have recently introduced for representing hyponymy in vector space (Erk, 2009). We now test its suitability as a general region model for word meaning. This model can be viewed as a prototype-style model that induces a region surrounding a central vector. As it does not record co-occurrences of feature values, we contrast it with a second model, an exemplar-style model using a k-nearest neighbor analysis, which can represent both degree of variance in each dimension and value co-occurrences.

Both models induce regions representations without labeled data. The idea on which both models are based is to use *word token* vectors to estimate a

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 57–65,
Boulder, Colorado, June 2009.

region representation. We evaluate the two region models on a task of token classification: Given a point in vector space, the task is predict the word of which it is a token vector.

By representing the meaning of words as regions in vector space, we can describe areas in which points encode similar meanings. This description is flexible, depending on the target word in question, rather than uniform for all words through a fixed distance threshold from the target's type vector. One possible application of region models of word meaning is in the task of determining the appropriateness of a paraphrase in a given context (Connor and Roth, 2007). This task is highly relevant for textual entailment (Szpektor et al., 2008). Current vector space approaches typically compare the target word's token vector to the type vector of the potential paraphrase (Mitchell and Lapata, 2008; Erk and Pado, 2008). A region model could instead test the target's token vector for inclusion in the potential paraphrase's region.

2 Related work

This section discusses existing vector space models and compares vector space models in computational linguistics to feature-based models of human concept representation in psychology.

Vector space models. Vector space models represent the meaning of a target word as a vector in a high-dimensional space (Lund and Burgess, 1996; Landauer and Dumais, 1997; Sahlgren and Karlgren, 2005; Padó and Lapata, 2007; Jones and Mewhort, 2007). Dimensions stand for context items which which the target word has been observed to co-occur, for example other words (Lund and Burgess, 1996) or syntactic paths (Padó and Lapata, 2007). In the simplest case, the value on a dimension is the raw co-occurrence count between the target word and the context item for which the dimension stands. Raw counts are often transformed, for example using a log-likelihood transformation (Lowe, 2001). Sometimes the vector space as a whole is transformed using dimensionality reduction (Landauer and Dumais, 1997).

In NLP, vector space models have featured most prominently in information retrieval (Manning et al., 2008), but have also been used for ontology learning (Lin, 1998; Snow et al., 2006; Gorman and Curran, 2006) and word sense-related tasks (McCarthy et al., 2004; Schütze, 1998). In psychology, vector space models have been used to model synonymy (Landauer and Dumais, 1997; Padó and Lapata, 2007), lexical priming phenomena (Lowe and McDonald, 2000), and similarity judgments (McDonald and Ramscar, 2001). There have also been studies on inducing hyponymy information from vector space representations. Geffet and Dagan (2005) use a dimension re-weighting scheme, then predict entailment when the most highly weighted dimensions of two verbs stand in a subset relation. However, they find that while recall of this method is good (whenever some senses of two words stand in an entailment relation, top-weighted dimensions of their vectors stand in a subset relation), precision is problematic. Weeds, Weir and McCarthy (2004) introduce the notion of distributional generality (x is more distributionally general than y if x occurs in more contexts than y) and find that for hyponym-hypernym pairs from WordNet, hyponyms are typically more distributionally general. (As they study only word pairs that are known to be related by hyponymy, they test for recall but not precision.) Erk (2009) suggests that while it may not be possible to *induce* hyponymy information from a vector space representation, it is possible to *encode* it in a vector space representation after it has been obtained through some other means.

Vector space models of word tokens. Vector space models have mostly been used to represent the meaning of a word *type* by summing its co-occurrence counts over a complete corpus. There are several approaches to computing vectors for individual word *tokens*. All of them compute word type vectors first, then combine them into token vectors. Kintsch (2001) and Mitchell and Lapata (2008) combine the target's type vector with that of a single word in the target's syntactic context. Landauer and Dumais (Landauer and Dumais, 1997) and Schütze (1998) combine the type vectors of all the words surrounding the target token. Erk and Padó (2008) combine the target's type vector with a vector representing the selectional preference of a single word in the target's syntactic context. Smolensky (1990) focuses on integrating syntactic information in the vector representation rather than

on representing the lexical meaning of the target.

Feature-based models of human concept representation. Many models of human concept representation in psychology are based on vectors of features (e.g. (Smith et al., 1988; Hampton, 1991; Nosofsky, 1986)). Features in these models are typically weighted to represent their importance to the concept in question. Similarity to a given feature vector is usually taken to decrease *exponentially* with distance from that vector, following Shepard's law (Shepard, 1987). Categorization involves competition between categories. Feature-based models of human concept representation can be broadly categorized into *prototype models*, which represent a concept by a single summary representation, and *exemplar models*, which assume that categorization is by comparison to remembered exemplars. As an example of a feature-based model of concept representation, we show the definition of Nosofsky's (1986) Generalized Context Model (GCM). This exemplar model estimates the probability of categorizing an exemplar $\vec{e}$ as a member of a concept C as

$$P(C|\vec{e}) = \frac{\sum_{\vec{\eta} \in C} w_{\vec{\eta}} sim(\vec{\eta}, \vec{e})}{\sum_{\text{concept } C'} \sum_{\vec{\eta} \in C'} w_{\vec{\eta}} sim(\vec{\eta}, \vec{e})} \quad (1)$$

where the concept C is a set of remembered exemplars, $w_{\vec{\eta}}$ is an exemplar weight, and the similarity $sim(\vec{\eta}, \vec{e})$ between $\vec{\eta}$ and $\vec{e}$ is defined as

$$sim(\vec{\eta}, \vec{e}) = exp(z \cdot \sum_{\text{dimension } i} w_i(\eta_i - e_i)^2) \quad (2)$$

Here, z is a general sensitivity parameter, w_i is a weight for dimension i, and η_i, e_i are the values of $\vec{\eta}$ and $\vec{e}$ on dimension i. This model shows all the properties listed above: It has weighted dimensions through the w_i. It incorporates Shepard's law through the exponential relation between sim and the sum of squared value distances $w_i(\eta_i - e_i)^2$. Competition between categories arises through the normalization of $\vec{e}$'s similarity to C by the similarity to all other categories in Eq. (1). While feature-based models of concept representation talk about concepts rather than word meaning, Murphy (2002) argues that there is "overwhelming empirical evidence for the conceptual basis of word meaning" through experimental results on conceptual phenomena that have also been shown to hold for words.

Gärdenfors (2004) proposes a model that represents concepts as convex regions in a *conceptual space*. Feature structures play no central role in this model, but Gärdenfors suggests that concepts may be represented by a central point, such that categorization could simply be determining the nearest central point (without positing an exponential relation between distance and similarity).

3 Models

In this section, we present two models for representing word meaning as regions in vector space.

The centered model. The first model that we define, which we call the *centered model*, is prototype-like. As the representation for a target word, it induces a region surrounding the target's type vector (Erk, 2009). Let w be the target word and $\vec{w}$ its type vector. Let $\vec{x}$ be a point in the same vector space. To predict whether $\vec{x}$ represents the same meaning as $\vec{w}$, we estimate the probability $P(\text{IN}(\vec{x}, \vec{w}))$ that $\vec{x}$ is in the region around $\vec{w}$, using a log-linear model:

$$P(\text{IN}(\vec{x}, \vec{w})) = \frac{1}{Z} \exp(\sum_i \beta_i^{\text{IN}} f_i(\vec{x}, \vec{w})) \quad (3)$$

where the f_i are features that characterize the point $\vec{x}$, and the β_i^{IN} are weights identifying the importance of the different features for the class IN. Z is a normalizing factor that ensures that P is a probability distribution: If $P(\text{OUT}(\vec{x}, \vec{w})) = 1 - P(\text{IN}(\vec{x}, \vec{w}))$ is the probability that $\vec{x}$ is not in the region around $\vec{w}$, with associated weights β_i^{OUT} for the same features f_i, then $Z = \sum_{\ell=\text{IN,OUT}} exp(\sum_i \beta_i^\ell f_i(\vec{x}, \vec{w}))$.

We define the features f_i as follows: If $\vec{w} = \langle w_1, \ldots, w_n \rangle$, we define the feature $f_i(\vec{x}, \vec{w})$, for $1 \leq i \leq n$, as the squared distance between $\vec{w}$ and $\vec{x}$ on dimension i:

$$f_i(\vec{x}) = (w_i - x_i)^2 \quad (4)$$

This model, like feature-based models of categorization from psychology, has weighted dimensions through the β_i. It follows Shepard's law – the exponential relation between similarity and distance – through the exponential function in Eq. (3). Competition between categories is implicit in the estimation of $P(\text{OUT}(\vec{x}, \vec{w}))$.

Most of the weights β_i^{IN} can reasonably be expected to be negative, since a negative β_i^{IN} indicates that membership of a point $\vec{x}$ in the w-region gets less likely as the distance $(w_i - x_i)^2$ increases. If β_i^{IN} has a large negative value, categorization is highly sensitive to changes in the ith dimension. If on the other hand, β_i^{IN} is negative but close to zero, this means that vector entries in dimension i can vary greatly without much influence on categorization.

The parameters β_i^{IN} and β_i^{OUT} need to be estimated from training data. Although the log-likelihood model is a supervised learning scheme, we do not need to take recourse to labeled data. Instead, we use token vectors: Token vectors of w will serve as positive training data for estimating $P(\text{IN}(\vec{x}, \vec{w}))$, and token vectors of other words than w will constitute negative training data. The amount of preprocessing needed depends on the approach to computing token vectors that we use. We will use an approach that combines w's type vector with that of a single word in its syntactic context. This presupposes a syntactic parse of the corpus. Note that we could just as well have used a Schütze-style approach, which does not rely on parsing.

The distributed model. The second model that we consider is an exemplar-style, instance-based model. The simplest instance-based models are k-nearest neighbor classifiers, which assign to a test item the majority label of its k nearest neighbors among the training items. We will here use a very simple model, doing k nearest neighbor classification where the distance between two vectors $\vec{w}$ and $\vec{x}$ is the sum of dimension distances δ_i with

$$\delta_i = \frac{\beta_i |w_i - x_i|}{\max_i - \min_i}$$

$\max_i$ and $\min_i$ are the maximum and minimum values observed for dimension i, and β_i is a feature weight. We use a standard feature weighting method, *gain ratio*, which is information gain normalized by the entropy of feature values. Information gain on its own has a bias towards features with many values, which gain ratio attenuates in favor of features with lower entropy:

$$\beta_i = \frac{H(C) - \sum_{y \in val(i)} P(y) H(C|y)}{-\sum_{y \in val(i)} P(y) \log_2 P(y)} \quad (5)$$

for the set $C = \{\text{IN}, \text{OUT}\}$ of classes and sets $val(i)$ of values seen for dimension i. We call this the *distributed model*. As with the centered model, we compare it to models of concept representation: It has weighted dimensions (Eq. (5)), and it incorporates competition between categories by storing both positive and negative exemplars and categorizing according to the majority among the k nearest neighbors. However, it does not implement Shepard's law. It additionally differs from the GCM (Eq. (1)) in basing categorization on the k nearest neighbors rather than summed similarity to all neighbors.

Like the centered model, the distributed model needs both positive and negative training data. Again, labeled data is not necessary as we can use word token vectors. Positive training data consists of tokens of the target word, and tokens of other words are negative training data. This model does not make use of the target's type vector.

Above we have discussed two pieces of information that region models can encode and that are hard to encode in single-point models of word meaning: *variance in each dimension* and *co-occurrence of feature values*. The centered model encodes the variance in the values of each dimension through the weights β_i^{IN}, but it does not retain information on feature values of different dimensions that tend to co-occur. The distributed model encodes both variance in each dimension and co-occurrence of feature values through the remembered exemplars. So the centered model should do well for monosemous words, since it seems reasonable that their token vectors should form a single region around the type vector. For polysemous words, token vectors could be more scattered in semantic space, in which case the distributed model should do better.

Note that neither the centered nor the distributed model is a clustering model: Both are supervised models learning the distinctions between tokens of the target word and other vectors. Neither of them groups vectors in an unsupervised fashion.

Hard versus soft region boundaries. In the current paper, we consider only regions with sharp boundaries. In the centered model, a point $\vec{x}$ will be considered a member of the w-region if $P(\text{IN}(\vec{x}, \vec{w})) \geq 0.5$. In the distributed model, $\vec{x}$ will be considered a member if the majority of its k nearest neighbors are members. However, it is im-

portant that both models can also be used to represent regions with soft boundaries. In the centered model, we can use $P(\text{IN}(\vec{x}, \vec{w}))$ without a threshold. In the distributed model, we can use the fraction of k that are positive instances, or we can compute summed similarity to the positive instances like the GCM does. So both models can be used to estimate degrees of membership in a target word's region.

4 Task, Data, and Implementation

This section describes the task used for evaluation, the data, and the implementation of the models.

Task. The main task will be for a model trained on a target word w to predict, for a given point $\vec{x}$ in semantic space, whether $\vec{x}$ is a token vector of w or not. This task is a direct test of whether the region induced for w succeeds in characterizing the region in semantic space in which tokens of w will occur.

As an example, consider the target word *supersede*: Region models of *supersede* will be trained on tokens of *supersede* in a training dataset. One such token is *supersede knowledge* (i.e., *knowledge* as the direct object of *supersede*). We compute a token vector for this occurrence by combining the type vectors of *supersede* and *knowledge*. After training a model, we test it on tokens occurring in a test dataset. Positive test items are tokens of *supersede*, and negative test items are tokens of other words, for example *guard*. An example of a positive test item is *supersede collection*. The test items will consist solely of tokens that do not occur in the training data.

Data. We focus on verbs in this paper since paraphrase appropriateness for verbs is an important task in the context of textual entailment. Since we suspect that the centered model will be better suited to modeling monosemous words while the distributed model should do equally well on monosemous and polysemous words, we first test a group of monosemous verbs, then a mixed group. We use WordNet 3.0 to form the two groups. The first group consists of all verbs listed in WordNet 3.0 as being monosemous. We refer to this set as ***Mon***. Since we also want to compare the two region models on the task of hyponymy encoding (Erk, 2009), we use as our set of mixed monosemous and polysemous verbs the verbs used there to test hyponymy encoding: the set of all verbs that are hypernyms of the *Mon* verbs according to WordNet 3.0. We call this set ***Hyp***.

We use the British National Corpus (BNC) to compute the vector space and as our source of target word tokens. We need token vectors for training the two region models, and we need separate, previously unseen token vectors as test data. So we split the written portion of the BNC in half at random, leaving files intact. This yielded a training and a test set. We computed word type vectors from the training half of the BNC, using a syntax-based vector space (Padó and Lapata, 2007) of 500 dimensions, with raw co-occurrence counts as dimension values. We used the `dv` package[1] to compute type vectors from a Minipar (Lin, 1993) parse of the BNC.

We computed token vectors by combining the target verb's type vector with the type vector of the word occurring as the target's direct object. We test three methods for combining type vectors: First, component-wise multiplication (below called ***mult***), which showed best results in Mitchell and Lapata's (2008) analysis. Second, component-wise averaging (below called ***avg***), a variant of type vector addition, a method often used for computing token vectors. Third, we consider component-wise minimum (***min***), which can be viewed as a kind of intersection of the contexts with which the two words have been observed. We used the training half of the BNC to extract training tokens of the target verbs, and the test half for extracting test tokens. We used only those verb/object pairs as test tokens that did not also occur in the training data.

We restricted the set of verbs to avoid data sparseness issues, using only verbs that occurred with at least 50 different direct objects in the training part of the BNC. The direct objects, in turn, were restricted to exclude overly rare and overly frequent (and thus potentially uninformative) items. We restricted the direct objects to those with no more than 6,500 and no less than 270 occurrences in $Mon \cup Hyp$. The resulting set *Mon* consisted of 120 verbs, and *Hyp* consisted of 430 verbs.

Model implementation. We implemented the centered model using the OpenNLP maxent package[2], and the distributed model using TiMBL[3] in the IB1 setting with $k = 5$ nearest neighbors. We use bi-

[1] `http://www.nlpado.de/~sebastian/dv.html`
[2] `http://maxent.sourceforge.net/`
[3] `http://ilk.uvt.nl/timbl/`

	centered			distributed		
	Prec	Rec	F	Prec	Rec	F
mult	**100**	73.2	84.5	29.4	47.5	36.3
avg	99.6	91.3	**95.3**	71.1	**99.9**	83.1
min	97.9	85.4	91.2	21.0	90.3	34.1

Table 1: Results: token classification for monosemous verbs. Random baseline: Prec 0.8, Rec 49.8, F 1.6.

nary models throughout, such that the classification task is always between IN and OUT. In training and testing, each token vector was presented to a model only once, ignoring the frequency of direct objects.

5 Experiments

This section reports on experiments that test the performance of the two region models of word meaning in vector space that we have presented in Sec. 3, the centered and the distributed model.

Experiment 1: Token classification for monosemous verbs

In the first experiment, we test whether the two region models can identify novel tokens of the monosemous verbs in *Mon*. The task is the one described in Sec. 4. We focus on monosemous verbs first because we suspect that the centered model should do better here than on polysemous verbs. Both models were trained using token vectors computed from the training half of the BNC. Token vectors of the target verb were treated as positive data, and token vectors of other verbs as negative data.[4] We used resampling to restrict the number of negative items used during training, using 3% of the negative items, randomly sampled.[5] We use for testing only those direct objects that do not also appear in the training data, yielding 6,339 positive and 1,396,552 negative test items summed over all target verbs. The case of *supersede* discussed in Sec. 4 is an example of a monosemous verb according to WordNet 3.0.

Table 1 summarizes precision, recall and F-score results. Both models easily beat the random baseline. The centered model shows better performance overall than the distributed one, and the *avg* method of computing token vector worked best for both models. The centered model has extremely high precision throughout, while the distributed model has better recall for conditions *avg* and *min*. Table 2 breaks down the results by the frequency of the target verb, measured in the number of different verb/object tokens in the training data.

		centered			distributed		
	freq.	Prec	Rec	F	Prec	Rec	F
mult	50-100	100	59.3	74.5	20.8	47.2	28.9
	100-200	100	89.4	94.4	57.4	49.7	53.2
	200-500	100	97.4	98.7	92.1	41.1	56.9
avg	50 - 100	99.5	86.6	92.6	61.6	99.8	76.2
	100-200	99.7	96.6	98.1	86.3	100	92.6
	200-500	100	100	100	99.1	100	99.6
min	50-100	100	82.9	90.6	17.9	92.6	30.1
	100-200	98.2	88.2	93.0	25.4	89.2	39.6
	200-500	86.4	90.3	88.3	42.9	80.0	55.9

Table 2: Results: token classification for monosemous verbs, by target frequency

	centered			distributed		
# senses	Prec	Rec	F	Prec	Rec	F
all	**100**	92.9	96.3	99.6	**99.8**	**99.7**
1	100	86.1	92.5	99.0	99.5	99.2
2-5	100	90.8	95.2	99.4	99.6	99.5
6-10	100	93.5	96.7	99.9	99.9	99.9
11-20	100	96.6	98.3	100	100	100
≥ 21	100	99.5	99.7	100	100	100

Table 3: Results: Token classification for polysemous verbs, *avg* token computation. Random baseline: Prec 8.2, Rec 50.4, F 14.0.

Experiment 2: Token classification for polysemous verbs

We now test how the centered and distributed models fare on the same task, but with a mixture of monosemous and polysemous verbs. We use the verbs in *Hyp*, which in WordNet 3.0 have on average 6.79 senses. For example, *follow* is a WordNet hypernym of the monosemous *supersede*. It has 24 senses, among them *comply* and *postdate*. Among its training tokens are *follow instruction* and *follow dinner*. The first is probably the *comply* sense of *follow*, the second the *postdate* sense. An example of a test token (i.e., occurring in the test but not the train-

[4] This simplification breaks down for 6 of the 120 verbs (5%), which are in fact synonyms. We consider this an acceptable level of noise.

[5] The number of 3% was determined on a development set constructed by further splitting the training set into training and development portion.

ing data) is *follow tea*. (If *tea* is *tea time*, this is also the *postdate* sense.)

We computed type vectors for the *Hyp* verbs and their objects from the training half of the BNC, and computed token vectors using the best method from Exp. 1, *avg*. Again, we use for testing only those tokens that do not also appear in the training data. Due to the larger amount of data, we used resampling in the training as well as the test data, using only a random 3% of negative tokens for testing. This yielded 25,736 positive and 670,630 negative test items.

Table 3 shows the results: The first line has the overall results, and the following lines break down the results by the number of senses each lemma has in WordNet 3.0.[6] Both models, centered and distributed, easily beat the random baseline. The centered model has comparable results for the *Hyp* as for the *Mon* verbs (cf. Table 1), while the distributed model has better results for this dataset, and better results than the centered model. The centered model shows a marked improvement in recall as the number of senses increases.

Experiment 3: Encoding hyponymy

We first proposed the centered model as a method for encoding hyponymy information in a vector space representation (Erk, 2009). Hyponymy information from another source, in this case WordNet, was encoded in a centered region representation of a target verb by using tokens of the verb itself as well as tokens from its direct hyponyms in training the model. Negative data consisted of training data tokens that were not occurrences of the target verb or its direct hyponyms. In the example of the verb *follow*, the positive training data would contain tokens of *follow* along with tokens of *supersede* and *guard*, another direct hyponym of *follow*. Negative training tokens would include, for example, tokens of the word *destroy*. The resulting centered model, in this case of *follow*, was then tested on previously unseen tokens, for example *guard purpose* (a token of a hyponym) and *destroy lawn* (a token of a non-hyponym), with the task of predicting whether they were tokens of direct hyponyms of *follow* or not.

[6]The one-sense items in Table 3 are a 43 verb subset of *Mon*. The reason for the difference in performance in comparison to Table 1 is unclear, as the two sets have similar distributions of lemma frequencies.

centered			distributed		
Prec	Rec	F	Prec	Rec	F
95.2	43.4	59.6	68.3	**58.6**	**63.1**

Table 4: Results: Identifying hyponyms based on extended hypernym representations, *avg* token computation. Random baseline: Prec 11.0, Rec 50.2, F 18.0

We now repeat this experiment with the distributed model. We use the *direct* hypernyms of the verbs in *Mon*, with the same frequency restrictions as above. We refer to this set of 273 verbs as ***DHyp***. We train one centered and one distributed model for each verb w in *DHyp*. Positive training tokens for training a model for a verb $w \in$ *DHyp* are tokens of w and of all sufficiently frequent children of w in WordNet 3.0. Negative training tokens are tokens of other verbs in *DHyp* and their children. We again sample a random 3% of the negative data during both training and testing.

Table 4 shows the results. Both models again beat the baseline. The distributed model shows slightly better results overall, while the centered model has by far the highest precision.

Discussion

Performance on monosemous verbs. For the monosemous verbs in Exp. 1, both models succeed in inducing regions that characterize tokens of a target word with high precision as well as high recall. The extremely high precision of the centered model shows that in general the region surrounding the type vector does not contain any tokens of other verbs than the target. Concerning the distributed model, it is to be expected that in *min*, and even more so in *mult*, dimension values will vary more than in *avg*; this could explain the huge difference between *avg* and the other two conditions for this model. It is interesting to note that the centered model achieves better precision, while the distributed model reaches higher recall. Maybe it will be possible in later models to combine their strengths. The breakdown by frequency bands in Table 2 shows that in *mult* and *avg*, the models get strictly better with more data, while *min* has a precision/recall tradeoff.

Performance on polysemous verbs. For the polysemous verbs in Exp. 2, like for the monosemous verbs in Exp. 1, both models show excellent per-

formance in distinguishing tokens of the target verb from tokens of other verbs.[7] The distributed model surpasses the centered one on this dataset. However, it is not clear that this is because the contiguous region that the centered model infers is inappropriate for polysemous verbs. After all the centered model, too, achieves better performance on this dataset than on *Mon*. The fact that results get better with the degree of polysemy, at first surprising, may indicate that the centered model draws an overly tight boundary around the type vector and that this boundary improves when token vectors differ more, and are at greater distance from the type vector, as should be the case for more polysemous lemmas. Another possible reason for the better performance of both models is that this dataset is larger and in particular provides a larger set of negative data.

Encoding external information in a region model. In the hyponymy encoding task in Exp. 3, both models successfully encode hyponymy information in vector space representations. The centered model manages to derive a high-precision region around the type vector, while the distributed model makes use of outliers in the training data to achieve higher recall.

Comparing region representations to point representations. We now compare the two region models to existing variants of point-based vector space models. Both region models have dimension weights, whose function is somewhat similar to that of log-likelihood or mutual information transformations of raw co-occurrence counts: to estimate the importance of each dimension for characterizing the target word in question. However, dimension weights in region models are computed based on token vectors, while all co-occurrence count transformations work on type vectors.

The distributed model additionally has the ability to represent typical co-occurrences of feature values because the training tokens are remembered in their entirety. The most similar mechanism in point-based vector space models is probably dimensionality reduction, which strives to find latent dimensions that explain most of the variance in the data. But again, dimensionality reduction uses type vectors while the distributed model stores token vectors, which can show more variance than the type vectors alone.

Applications of region models. Region models of word meaning are interesting for the task of testing the appropriateness of paraphrases in context. Previous models either used competition between paraphrase candidates or a global similarity threshold to decide whether to accept a paraphrase candidate (Mitchell and Lapata, 2008; Szpektor et al., 2008). A region model of word meaning used for the same task would still require a threshold, in this case a threshold on membership probability, but the regions for which membership is tested could differ in their size, and the extent of each region would be learned individually from the data. To use the model, for example to test whether *trickle* is a good paraphrase for *run* in *the color ran*, we would test whether the sentence-specific token vector for *run* falls into the region of *trickle*.

6 Conclusion and outlook

In this paper, we have proposed using region models for word meaning in vector space, predicting regions in space in which points can be assumed to carry the same meaning. We have studied two models, the prototype-like *centered* models and the exemplar-like *distributed* model, both of which are learned without labeled data by making use of *token vectors* of the target word in question. Both models show excellent performance, with F-scores of 83%-99%, on the task of identifying previously unseen occurrences of the target word.

Our aim is to to test the usability of region models for predicting paraphrase appropriateness in context. The next step towards that will be to test region models on the task of identifying synonym tokens.

Acknowledgements. Many thanks to Jason Baldridge, David Beaver, Graham Katz, Alexander Koller, Ray Mooney, and Manfred Pinkal for helpful discussions. This work was supported by the Morris Memorial Grant from the New York Community Trust.

References

M. Connor and D. Roth. 2007. Context sensitive paraphrasing with a single unsupervised classifier. In *Proceedings of ECML-07*, Warsaw, Poland.

[7] The near-perfect performance in particular of the distributed model has been confirmed on a separate noun dataset.

K. Erk and S. Pado. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP-08*, Hawaii.

K. Erk. 2009. Supporting inferences in semantic space: representing words as regions. In *Proceedings of IWCS-8*, Tilburg, Netherlands.

P. Gärdenfors. 2004. *Conceptual spaces*. MIT press, Cambridge, MA.

M. Geffet and I. Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of ACL-05*, Ann Arbor, MI.

J. Gorman and J. R. Curran. 2006. Scaling distributional similarity to large corpora. In *Proceedings of ACL '06*, Sydney.

J. A. Hampton. 1991. The combination of prototype concepts. In P. Schwanenflugel, editor, *The psychology of word meanings*. Lawrence Erlbaum Associates.

P. Hanks. 2000. Do word meanings exist? *Computers and the Humanities*, 34(1-2):205–215(11).

M. Jones and D. Mewhort. 2007. Representing word menaing and order information in a composite holographic lexicon. *Psychological Review*, 114:1–37.

A. Kilgarriff. 1997. I don't believe in word senses. *Computers and the Humanities*, 31(2):91–113.

W. Kintsch. 2001. Predication. *Cognitive Science*, 25:173–202.

T. Landauer and S. Dumais. 1997. A solution to Platos problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

D. Lin. 1993. Principle-based parsing without overgeneration. In *Proceedings of ACL'93*, Columbus, Ohio.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *COLING-ACL98*, Montreal, Canada.

W. Lowe and S. McDonald. 2000. The direct route: Mediated priming in semantic space. In *Proceedings of the Cognitive Science Society*.

W. Lowe. 2001. Towards a theory of semantic space. In *Proceedings of the Cognitive Science Society*.

K. Lund and C. Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, 28:203—208.

C. D. Manning, P. Raghavan, and H. Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. 2004. Finding predominant senses in untagged text. In *Proceedings of ACL'04*, Barcelona, Spain.

S. McDonald and M. Ramscar. 2001. Testing the distributional hypothesis: The influence of context on judgements of semantic similarity. In *Proceedings of the Cognitive Science Society*.

J. Mitchell and M. Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08*, Columbus, OH.

G. L. Murphy. 2002. *The Big Book of Concepts*. MIT Press.

R. M. Nosofsky. 1986. Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology: General*, 115:39–57.

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

M. Sahlgren and J. Karlgren. 2005. Automatic bilingual lexicon acquisition using random indexing of parallel corpora. *Journal of Natural Language Engineering, Special Issue on Parallel Texts*, 11(3).

H. Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1).

R. Shepard. 1987. Towards a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323.

E. E. Smith, D. Osherson, L. J. Rips, and M. Keane. 1988. Combining prototypes: A selective modification model. *Cognitive Science*, 12(4):485–527.

P. Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:159–216.

R. Snow, D. Jurafsky, and A. Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING/ACL'06*.

I. Szpektor, I. Dagan, R. Bar-Haim, and J. Goldberger. 2008. Contextual preferences. In *Proceedings of ACL-08*, Columbus, OH.

J. Weeds, D. Weir, and D. McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of COLING-04*, Geneva, Switzerland.

Interactive Feature Space Construction using Semantic Information

Dan Roth and **Kevin Small**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{danr,ksmall}@illinois.edu

Abstract

Specifying an appropriate feature space is an important aspect of achieving good performance when designing systems based upon learned classifiers. Effectively incorporating information regarding semantically related words into the feature space is known to produce robust, accurate classifiers and is one apparent motivation for efforts to automatically generate such resources. However, naive incorporation of this semantic information may result in poor performance due to increased ambiguity. To overcome this limitation, we introduce the *interactive feature space construction* protocol, where the learner identifies inadequate regions of the feature space and in coordination with a domain expert adds descriptiveness through existing semantic resources. We demonstrate effectiveness on an entity and relation extraction system including both performance improvements and robustness to reductions in annotated data.

1 Introduction

An important natural language processing (NLP) task is the design of learning systems which perform well over a wide range of domains with limited training data. While the NLP community has a long tradition of incorporating linguistic information into statistical systems, machine learning approaches to these problems often emphasize learning sophisticated models over simple, mostly lexical, features. This trend is not surprising as a primary motivation for machine learning solutions is to reduce the manual effort required to achieve state of the art performance. However, one notable advantage of discriminative classifiers is the capacity to encode arbitrarily complex features, which partially accounts for their popularity. While this flexibility is powerful, it often overwhelms the system designer causing them to resort to simple features. This work presents a method to partially automate feature engineering through an interactive learning protocol.

While it is widely accepted that classifier performance is predicated on feature engineering, designing good features requires significant effort. One underutilized resource for descriptive features are existing *semantically related word lists* (SRWLs), generated both manually (Fellbaum, 1998) and automatically (Pantel and Lin, 2002). Consider the following named entity recognition (NER) example:

> *His father was rushed to [Westlake Hospital]*$_{\mathbf{ORG}}$*, an arm of [Resurrection Health Care]*$_{\mathbf{ORG}}$*, in west suburban [Chicagoland]*$_{\mathbf{LOC}}$*.*

For such tasks, it is helpful to know that *west* is a member of the SRWL *[Compass Direction]* and other such designations. If extracting features using this information, we would require observing only a subset of the SRWL in the data to learn the corresponding parameter. This statement suggests that one method for learning robust classifiers is to incorporate semantic information through features extracted from the more descriptive representation:

> *His father was rushed to Westlake [Health Care Institution], an [Subsidiary] of Resurrection Health Care, [Locative Preposition] [Compass Direction] suburban Chicagoland.*

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 66–74,
Boulder, Colorado, June 2009.

Deriving discriminative features from this representation often results in more informative features and a correspondingly simpler classification task. Although effective approaches along this vein have been shown to induce more accurate classifiers (Boggess et al., 1991; Miller et al., 2004; Li and Roth, 2005), naive approaches may instead result in higher sample complexity due to increased ambiguity introduced through these semantic resources. Features based upon SRWLs must therefore balance the tradeoff between descriptiveness and noise.

This paper introduces the *interactive feature space construction* (IFSC) protocol, which facilitates coordination between a domain expert and learning algorithm to interactively define the feature space during training. This paper describes the particular instance of the IFSC protocol where semantic information is introduced through abstraction of lexical terms in the feature space with their SRWL labels. Specifically, there are two notable contributions of this work: (1) an interactive method for the expert to directly encode semantic knowledge into the feature space with minimal effort and (2) a querying function which uses both the current state of the learner and properties of the available SRWLs to select informative instances for presentation to the expert. We demonstrate the effectiveness of this protocol on an entity and relation extraction task in terms of performance and labeled data requirements.

2 Preliminaries

Following standard notation, let $x \in \mathcal{X}$ represent members of an input domain and $y \in \mathcal{Y}$ represent members of an output domain where a learning algorithm uses a training sample $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{m}$ to induce a prediction function $h : \mathcal{X} \rightarrow \mathcal{Y}$. We are specifically interested in discriminative classifiers which use a feature vector generating procedure $\Phi(x) \rightarrow \mathbf{x}$, taking an input domain member x and generating a feature vector $\mathbf{x}$. We further assume the output assignment of h is based upon a scoring function $f : \Phi(\mathcal{X}) \times \mathcal{Y} \rightarrow \mathbb{R}$ such that the prediction is stated as $\hat{y} = h(x) = \text{argmax}_{y' \in \mathcal{Y}} f(\mathbf{x}, y')$.

The feature vector generating procedure is composed of a vector of feature generation functions (FGFs), $\Phi(x) = \langle \Phi_1(x), \Phi_2(x), \ldots, \Phi_n(x) \rangle$, where each feature generation function, $\Phi_i(x) \rightarrow \{0, 1\}$, takes the input x and returns the appropriate feature vector value. Consider the text "*in west suburban Chicagoland*" where we wish to predict the entity classification for *Chicagoland*. In this case, example active FGFs include $\Phi_{text=Chicagoland}$, $\Phi_{isCapitalized}$, and $\Phi_{text(-2)=west}$ while FGFs such as $\Phi_{text=and}$ would remain inactive. Since we are constructing sparse feature vectors, we use the *infinite attribute model* (Blum, 1992).

Semantically related word list (SRWL) feature abstraction begins with a set of variable sized word lists $\{\mathcal{W}\}$ such that each member lexical element (i.e. word, phrase) has at least one sense that is semantically related to the concept represented by $\mathcal{W}$ (e.g. $\mathcal{W}_{compass\_direction} = north, east, \ldots, southwest$). For the purpose of feature extraction, whenever the sense of a lexical element associated with a particular $\mathcal{W}$ appears in the corpus, it is replaced by the name of the corresponding SRWL. This is equivalent to defining a FGF for the specified $\mathcal{W}$ which is a disjunction of the functionally related FGFs over the member lexical elements (e.g. $\Phi_{text \in \mathcal{W}_{compass\_direction}} = \Phi_{text=north} \vee \Phi_{text=east} \vee \ldots \vee \Phi_{text=southwest}$).

3 Interactive Feature Space Construction

The machine learning community has become increasingly interested in protocols which allow interaction with a domain expert during training, such as the active learning protocol (Cohn et al., 1994). In active learning, the learning algorithm reduces the labeling effort by using a querying function to incrementally select unlabeled examples from a data source for annotation during learning. By carefully selecting examples for annotation, active learning maximizes the quality of inductive information while minimizing label acquisition cost.

While active learning has been shown to reduce sample complexity, we contend that it significantly underutilizes the domain expert – particularly for complex annotation tasks. More precisely, when a domain expert receives an instance, world knowledge is used to reason about the instance and supply an annotation. Once annotated and provided for training, the learner must recover this world knowledge and incorporate it into its model from a small number of instances, exclusively through induction.

Learning algorithms generally assume that the feature space and model are specified before learning begins and remain static throughout learning, where training data is exclusively used for parameter estimation. Conversely, the *interactive feature space construction* (IFSC) protocol relaxes this static feature space assumption by using information about the current state of the learner, properties of knowledge resources (e.g. SRWLs, gazetteers, unlabeled data, etc.), and access to the domain expert during training to interactively improve the feature space. Whereas active learning focuses on the labeling effort, IFSC reduces sample complexity and improves performance by modifying the underlying representation to simplify the overall learning task.

The IFSC protocol for SRWL abstraction is presented in Algorithm 1. Given a labeled data set $\mathcal{S}$, an initial feature vector generating procedure Φ_0, a querying function $\mathcal{Q} : \mathcal{S} \times h \rightarrow \mathcal{S}_{select}$, and an existing set of semantically related word lists, $\{\mathcal{W}\}$ (line 1), an initial hypothesis is learned (line 3). The querying function scores the labeled examples and selects an instance for interaction (line 6). The expert selects lexical elements from this instance for which feature abstractions may be performed (line 8). If the expert doesn't deem any elements viable for interaction, the algorithm returns to line 5. Once lexical elements are selected for interaction, the SRWL $\mathcal{W}_\epsilon$ associated with each selected element is retrieved (line 11) and refined by the expert (line 12). Using the validated SRWL definition $\mathcal{W}^*_\epsilon$, the lexical FGFs are replaced with the SRWL FGF (line 14). This new feature vector generating procedure Φ_{t+1} is used to train a new classifier (line 18) and the algorithm is repeated until the annotator halts.

Algorithm 1 Interactive Feature Space Construction

1: **Input:** Labeled training data $\mathcal{S}$, feature vector generating procedure Φ_0, querying function $\mathcal{Q}$, set of known SRWLs $\{W\}$, domain expert $\mathcal{A}^*$

2: $t \leftarrow 0$
3: $h_t \leftarrow \mathcal{A}(\Phi_t, \mathcal{S})$; learn initial hypothesis
4: $\mathcal{S}_{selected} \leftarrow \emptyset$
5: **while** annotator is willing **do**
6: $\quad \mathcal{S}_{select} \leftarrow \mathcal{Q}(\mathcal{S} \backslash \mathcal{S}_{selected}, h_t)$; $\mathcal{Q}$ proposes (labeled) instance for interaction
7: $\quad \mathcal{S}_{selected} \leftarrow \mathcal{S}_{selected} \cup \mathcal{S}_{select}$; mark selected examples to prevent reselection
8: $\quad E_{select} \leftarrow \mathcal{A}^*(\mathcal{S}_{select})$; the expert selects lexical elements for semantic abstraction
9: $\quad \Phi_{t+1} \leftarrow \Phi_t$; initialize new FGF vector with existing FGFs
10: $\quad$ **for each** $\epsilon \in E_{select}$ **do**
11: $\quad\quad$ Retrieve word list $\mathcal{W}_\epsilon$
12: $\quad\quad \mathcal{W}^*_\epsilon \leftarrow \mathcal{A}^*(\mathcal{W}_\epsilon)$; the expert refines the existing semantic class $\mathcal{W}_\epsilon$ for this task
13: $\quad\quad$ **for each** $\Phi \sim \epsilon$ **do**
14: $\quad\quad\quad \Phi_{t+1} \leftarrow (\Phi_{t+1} \backslash \Phi) \cup \Phi_{\mathcal{W}^*_\epsilon}$; replace features with SRWL features (e.g. $\Phi_{text=\epsilon} \rightarrow \Phi_{text \in \mathcal{W}^*_\epsilon}$)
15: $\quad\quad$ **end for**
16: $\quad$ **end for**
17: $\quad t \leftarrow t + 1$
18: $\quad h_t \leftarrow \mathcal{A}(\Phi_t, \mathcal{S})$; learn new hypothesis
19: **end while**

20: **Output:** Learned hypothesis h_T, final feature space Φ_T, refined semantic classes $\{\mathcal{W}^*\}$

3.1 Method of Expert Interaction

The method of interaction for active learning is very natural; data annotation is required regardless. To increase the bandwidth between the expert and learner, a more sophisticated interaction must be allowed while ensuring that the expert task of remains reasonable. We require the interaction be restricted to mouse clicks. When using this protocol to incorporate semantic information, the primary tasks of the expert are (1) selecting lexical elements for SRWL feature abstraction and (2) validating membership of the SRWL for the specified application.

3.1.1 Lexical Feature Selection (Line 8)

Once an instance is selected by the querying function (line 6), the the domain expert selects lexical elements (i.e. words, phrases) believed appropriate for SRWL feature abstraction. This step is summarized by Figure 1 for the example introduced in Section 1.

For this NER example, features extracted include the words and bigrams which form the named entity and those within a surrounding two word window. All lexical elements which have membership to at least one SRWL and are used for feature extraction are marked with a box and may be selected by the user for interaction. In this particular case, the system has made a mistake in classification of

His father was rushed *to [Westlake* Hospital *]*$_{\mathbf{ORG}}$*, an* arm *of [Resurrection* Health Care *]*$_{\mathbf{ORG}}$*, in* west suburban *[Chicagoland]*$_{\mathbf{ORG}}$.

Figure 1: Lexical Feature Selection – All lexical elements with SRWL membership used to derive features are boxed. Elements used for the incorrect prediction for *Chicagoland* are double-boxed. The expert may select any boxed element for SRWL validation.

Chicagoland and the lexical elements used to derive features for this prediction are emphasized with a double-box for expository purposes. The expert selects lexical elements which they believe will result in good feature abstractions; the querying function must present examples believed to have high impact.

3.1.2 Word List Validation (Lines 11 &12)

Once the domain expert has selected a lexical element for SRWL feature abstraction, they are presented with the SRWL $\mathcal{W}$ to validate membership for the target application as shown in Figure 2. In this particular case, the expert has chosen to perform two interactions, namely for the lexical elements *west* and *suburban*. Once they have chosen which words and phrases will be included in this particular feature abstraction, $\mathcal{W}$ is updated and the associated features are replaced with their SRWL counterpart. For example, $\Phi_{text=west}$, $\Phi_{text=north}$, etc. would all be replaced with $\Phi_{text \in \mathcal{W}_{A1806}}$ later in lines 13 & 14.

A1806: southeast, northeast, south
southeast, northeast, south, north, southwest, west, east, northwest, ~~inland~~, ~~outside~~

A1558: suburban, ~~nearby~~, downtown
suburban, ~~nearby~~, downtown, urban, metropolitan, ~~neighboring~~, ~~near~~, coastal

Figure 2: Word List Validation – Completing two domain expert interactions. Upon selecting either double-boxed element in Figure 1, the expert validates the respective SRWL for feature extraction.

Accurate sense disambiguation is helpful for effective SRWL feature abstraction to manage situations where lexical elements belong to multiple lists. In this work, we first disambiguate by predicted part of speech (POS) tags. In cases of multiple SRWL senses for a POS, the given SRWLs (Pantel and Lin, 2002) rank list elements according their semantic representativeness which we use to return the highest ranked sense for a particular lexical element. Also, as SRWL resources emphasize recall over precision, we reduce expert effort by using the Google n-gram counts (Brandts and Franz, 2006) to automatically prune SRWLs.

3.2 Querying Function (Line 6)

A primary contribution of this work is designing an appropriate querying function. In doing so, we look to maximize the impact of interactions while minimizing the total number. Therefore, we look to select instances for which (1) the current hypothesis indicates the feature space is insufficient and (2) the resulting SRWL feature abstraction will help improve performance. To account for these two somewhat orthogonal goals, we design two querying functions and aggregate their results.

3.2.1 Hypothesis-Driven Querying

To find areas of the feature space which are believed to require more descriptiveness, we look to emphasize those instances which will result in the largest updates to the hypothesis. To accomplish this, we adopt an idea from the active learning community and score instances according to their margin relative to the current learned hypothesis, $\rho(f_t, x_i, y_i)$ (Tong and Koller, 2001). This results in the *hypothesis-driven* querying function

$$\mathcal{Q}_{margin} = \underset{i=1,\ldots,m}{\operatorname{argsort}} \rho(f_t, x_i, y_i)$$

where the $\operatorname{argsort}$ operator is used to sort the input elements in ascending order (for multiple instance selection). Unlike active learning, where selection is from an unlabeled data source, the quantity of labeled data is fixed and labeled data is selected during each round. Therefore, we use the true margin and not the expected margin. This means that we will first select instances which have large mistakes, followed by those instances with small mistakes, and finally instances that make correct predictions in the order of their confidence.

3.2.2 SRWL-Driven Querying

An equally important goal of the querying function is to present examples which will result in SRWL feature abstractions of broad usability. Intuitively, there are two criteria distinguishing desirable SRWLs for this purpose. First of all, large lists are desirable as there are many lists of cities, countries, corporations, etc. which are extremely informative. Secondly, preference should be given to lists where the distribution of lexical elements within a particular word list, $\epsilon \in \mathcal{W}$, is more uniform. For example, consider $\mathcal{W} = \{devour, feed\_on, eat, consume\}$. While all of these terms belong to the same SRWL, learning features based on *eat* is sufficient to cover most examples. To derive a *SRWL-driven* querying function based on these principles, we use the word list entropy, $H(\mathcal{W}) = -\sum_{\epsilon \in \mathcal{W}} p(\epsilon) \log p(\epsilon)$. The querying score for a sentence is determined by its highest entropy lexical element used for feature extraction, resulting in the querying function

$$\mathcal{Q}_{entropy} = \underset{i=1,\ldots,m}{\text{argsort}} \left[\underset{\epsilon \sim \Phi_{x_i}}{\text{argmin}} -H(\mathcal{W}_\epsilon) \right]$$

This querying function is supported by the underlying assumption of SRWL abstraction is that there exists a true feature space $\Phi^*(x)$ which is built upon SRWLs and lexical elements but is being approximated by $\Phi(x)$, which doesn't use semantic information. In this context, a lexical feature provides one bit of information to the prediction function while a SRWL feature provides information content proportional to its SRWL entropy $H(\mathcal{W})$.

To study one aspect of this phenomena empirically, we examine the rate at which words are first encountered in our training corpus from Section 4, as shown by Figure 3. The first observation is the usefulness of SRWL feature abstraction in general as we see that when including an entire SRWL from (Pantel and Lin, 2002) whenever the first element of the list is encountered, we cover the unigram vocabulary much more rapidly. The second observation is that when sentences are presented in the order of the average SRWL entropy of their words, this coverage rate is further accelerated. Figure 3 helps explain the recall focused aspect of SRWL abstraction while we rely on hypothesis-driven querying to target interactions for the specific task at hand.

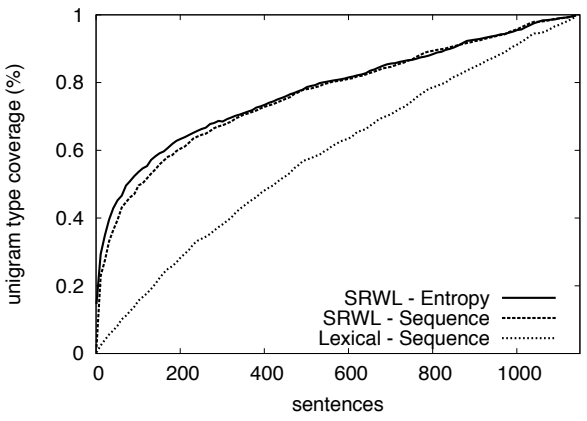


Figure 3: The Impact of SRWL Abstraction and SRWL-driven Querying – The first occurrence of words occur at a much lower rate than the first occurrence of words when abstracted through SRWLs, particularly when sentences are introduced as ranked by average SRWL entropy calculated using (Brandts and Franz, 2006).

3.2.3 Aggregating Querying Functions

To combine these two measures, we use the Borda count method of rank aggregation (Young, 1974) to find a consensus between the two querying functions without requiring calibration amongst the actual ranking scores. Defining the rank position of an instance by $r(x)$, the Borda count based querying function is stated by

$$\mathcal{Q}_{Borda} = \underset{i=1,\ldots,m}{\text{argsort}} [r_{margin}(x_i) + r_{entropy}(x_i)]$$

$\mathcal{Q}_{Borda}$ selects instances which consider both wide applicability through $r_{entropy}$ and which focus on the specific task through r_{margin}.

4 Experimental Evaluation

To demonstrate the IFSC protocol on a practical application, we examine a three-stage pipeline model for entity and relation extraction, where the task is decomposed into sequential stages of segmentation, entity classification, and relation classification (Roth and Small, 2008). Extending the standard classification task, a pipeline model decomposes the overall classification into a sequence of D stages such that each stage $d = 1, \ldots, D$ has access to the input instance along with the classifications from all previous stages, $\hat{y}^{(d)}$. Each stage of the pipeline model uses a feature vector generating procedure

$\Phi^{(d)}(x, \hat{y}^{(0)}, \ldots, \hat{y}^{(d-1)}) \rightarrow \mathbf{x}^{(d)}$ to learn a hypothesis $h^{(d)}$. Once each stage of the pipelined classifier is learned, predictions are made sequentially, where

$$\hat{\mathbf{y}} = h(x) = \left\langle \underset{y' \in \mathcal{Y}^{(d)}}{\operatorname{argmax}} f^{(d)}\left(\mathbf{x}^{(d)}, y'\right) \right\rangle_{d=1}^{D}$$

Each pipeline stage requires a classifier which makes multiple interdependent predictions based on input from multiple sentence elements $x \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_{n_x}$ using a structured output space, $y^{(d)} \in \mathcal{Y}_1^{(d)} \times \cdots \times \mathcal{Y}_{n_y}^{(d)}$. More specifically, segmentation makes a prediction for each sentence word over $\mathcal{Y} \in \{begin, inside, outside\}$ and constraints are enforced between predictions to ensure that an *inside* label can only follow a *begin* label. Entity classification begins with the results of the segmentation classifier and classifies each segment into $\mathcal{Y} \in \{person, location, organization\}$. Finally, relation classification labels each predicted entity pair with $\mathcal{Y} \in \{located\_in, work\_for, org\_based\_in, live\_in, kill\} \times \{left, right\} + no\_relation$.

The data used for empirical evaluation was taken from (Roth and Yih, 2004) and consists of 1436 sentences, which is split into a 1149 (80%) sentence training set and a 287 (20%) sentence testing set such that all have at least one active relation. SRWLs are provided by (Pantel and Lin, 2002) and experiments were conducted using a custom graphical user interface (GUI) designed specifically for the IFSC protocol. The learning algorithm used for each stage of the classification task is a regularized variant of the structured Perceptron (Collins, 2002). Resources used to perform experiments are available at `http://L2R.cs.uiuc.edu/~cogcomp/`.

We extract features in a method similar to (Roth and Small, 2008), except that we do not include gazetteer features in $\Phi_0^{(d)}$ as we will include this type of external information interactively. Secondly, we use SRWL features as introduced. The segmentation features include the word/SRWL itself along with the word/SRWL of three words before and two words after, bigrams of the word/SRWL surrounding the word, capitalization of the word, and capitalization of its neighbor on each side. Entity classification uses the segment size, the word/SRWL members within the segment, and a window of two word/SRWL elements on each side. Relation classification uses the same features as entity classification along with the entity labels, the length of the entities, and the number of tokens between them.

4.1 Interactive Querying Function

When using the interactive feature space construction protocol for this task, we require a querying function which captures the hypothesis-driven aspect of instance selection. We observed that basing $\mathcal{Q}_{margin}$ on the relation stage performs best, which is not surprising given that this stage makes the most mistakes, benefits the most from semantic information, and also has many features which are similar to features from previous stages. Therefore, we adapt the querying function described by (Roth and Small, 2008) for the relation classification stage and define our margin for the purposes of instance selection as

$$\rho_{relation} = \min_{i=1,\ldots,n_y} \left[f_{y_+}(\mathbf{x}, i) - f_{\dot{y}_+}(\mathbf{x}, i) \right]$$

where $\dot{y} = \operatorname{argmax}_{y' \in \mathcal{Y} \setminus y} f_{y'}(\mathbf{x})$, the highest scoring class which is not the true label, and $\mathcal{Y}_+ = \mathcal{Y} \setminus no\_relation$.

4.2 Interactive Protocol on Entire Data Set

The first experiments we conduct uses all available training data (i.e. $|\mathcal{S}| = 1149$) to examine the improvement achieved with a fixed number of IFSC interactions. A single interaction is defined by the expert selecting a lexical element from a sentence presented by the querying function and validating the associated word list. Therefore, it is possible that a single sentence may result in multiple interactions.

The results for this experimental setup are summarized in Table 1. For each protocol configuration, we report F1 measure for all three stages of the pipeline. As our simplest baseline, we first train using the default feature set without any semantic features (**Lexical Features**). The second baseline is to replace all instances of any lexical element with its SRWL representation as provided by (Pantel and Lin, 2002) (**Semantic Features**). The next two baselines attempt to automatically increase precision by defining each semantic class using only the top fraction of the elements in each SRWL (**Pruned Semantic (top {1/2,1/4})**). This pruning procedure often results in smaller SRWLs with a more precise specification of the semantic concept.

	Lexical Features	Semantic Features	Pruned Semantic (top 1/2)	Pruned Semantic (top 1/4)	50 interactions Interactive (select only)	50 interactions Interactive (select & validate)
Segmentation	90.23	90.14	90.77	89.71	92.24	**93.43**
Entity Class.	82.17	83.28	83.93	83.04	85.81	**88.76**
Relation Class.	54.67	55.20	56.34	56.21	59.14	**62.08**

Table 1: Relative performance of the stated experiments conducted over the entire available dataset. The interactive feature construction protocol outperforms all non-interactive baselines, particularly for later stages of the pipeline while requiring only 50 interactions.

Finally, we consider the interactive feature space construction protocol at two different stages. We first consider the case where 50 interactions are performed such that the algorithm assumes $\mathcal{W}^* = \mathcal{W}$, that is, the expert selects features for abstraction, but doesn't perform validation (**Interactive (select only)**). The second experiment performs the entire protocol, including validation (**Interactive (select & validate)**) for 50 interactions. On the relation extraction task, we observe a 13.6% relative improvement over the lexical model and a 10.2% relative improvement over the best SRWL baseline F1 score.

4.3 Examination of the Querying Function

As stated in section 3.2, an appropriate querying function presents sentences which will result in the expert selecting features from that example and for which the resulting interactions will result in a large performance increase. The former is difficult to model, as it is dependent on properties of the sentence (such as length), will differ from user to user, and anecdotally is negligibly different for the three querying functions for earlier interactions. However, we are able to measure the performance improvement of interactions associated with different querying functions. For our second experiment, we evaluate the relative performance of the three querying functions defined after every ten interactions in terms of the F1 measure for relation extraction. The results of this experiment are shown in figure 4, where we first see that the $\mathcal{Q}_{random}$ generally leads to the least useful interactions. Secondly, while $\mathcal{Q}_{entropy}$ performs well early, $\mathcal{Q}_{margin}$ works better as more interactions are performed. Finally, we also observe that $\mathcal{Q}_{Borda}$ exceeds the performance envelope of the two constituent querying functions.

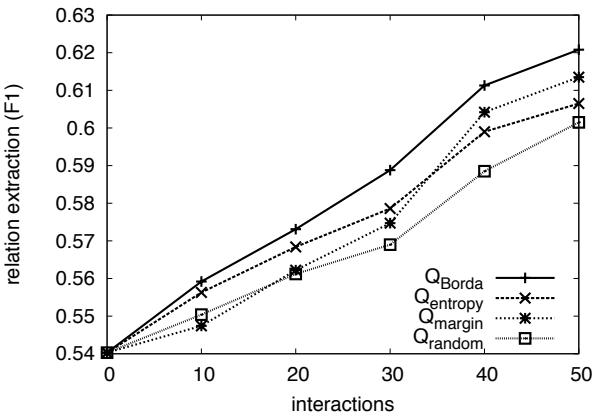


Figure 4: Relative performance of interactions generated through the respective querying functions. We see that $\mathcal{Q}_{entropy}$ performs well for a small number of interactions, $\mathcal{Q}_{margin}$ performs well as more interactions are performed and $\mathcal{Q}_{Borda}$ outperforms both consistently.

4.4 Robustness to Reduced Annotation

The third set of experiments consider the relative performance of the configurations from the first set of experiments as the amount of available training data is reduced. To study this scenario, we perform the same set of experiments with 50 interactions while varying the size of the training set (e.g. $|\mathcal{S}| = \{250, 500, 600, 675, 750, 1000\}$), summarizing the results in Figure 5. One observation is that the interactive feature space construction protocol outperforms all other configurations at all annotation levels. A second important observation is made when comparing these results to those presented in (Roth and Small, 2008), where this data is labeled using active learning. In (Roth and Small, 2008), once 65% of the labeled data is observed, a performance level is achieved comparable to training on the entire labeled dataset. In this work, an interpo-

lation of the performance at 600 and 675 labeled instances implies that we achieve a performance level comparable to training on all of the data of the baseline learner while about 55% of the labeled data is observed at random. Furthermore, as more labeled data is introduced, the performance continues to improve with only 50 interactions. This supports the hypothesis that a good representation is often more important than additional training data, even when the data is carefully selected.

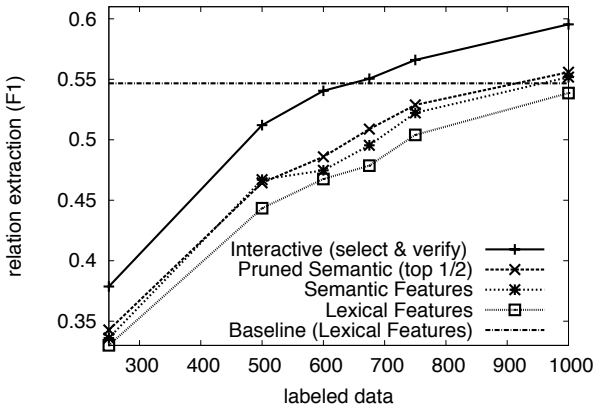


Figure 5: Relative performance of several baseline algorithm configurations and the interactive feature space construction protocol with variable labeled dataset sizes. The interactive protocol outperforms other baseline methods in all cases. Furthermore, the interactive protocol **(Interactive)** outperforms the baseline lexical system **(Baseline)** trained on all 1149 sentences even when trained with a significantly smaller subset of labeled data.

5 Related Work

There has been significant recent work on designing learning algorithms which attempt to reduce annotation requirements through a more sophisticated annotation method. These methods allow the annotator to directly specify information about the feature space in addition to providing labels, which is then incorporated into the learning algorithm (Huang and Mitchell, 2006; Raghavan and Allan, 2007; Zaidan et al., 2007; Druck et al., 2008; Zaidan and Eisner, 2008). Additionally, there has been recent work using explanation-based learning techniques to encode a more expressive feature space (Lim et al., 2007). Amongst these works, the only interactive learning protocol is (Raghavan and Allan, 2007) where instances are presented to an expert and features are labeled which are then emphasized by the learning algorithm. Thus, in this case, although additional information is provided the feature space itself remains static. To the best of our knowledge, this is the first work that interactively modifies the feature space by abstracting the FGFs.

6 Conclusions and Future Work

This work introduces the *interactive feature space construction* protocol, where the learning algorithm selects examples for which the feature space is believed to be deficient and uses existing semantic resources in coordination with a domain expert to abstract lexical features with their SRWL names. While the power of SRWL abstraction in terms of sample complexity is evident, incorporating this information is fraught with pitfalls regarding the introduction of additional ambiguity. This interactive protocol finds examples for which the domain expert will recognize promising semantic abstractions and for which those semantic abstraction will significantly improve the performance of the learner. We demonstrate the effectiveness of this protocol on a named entity and relation extraction system.

As a relatively new direction, there are many possibilities for future work. The most immediate task is effectively quantifying interaction costs with a user study, including the impact of including users with varying levels of expertise. Recent work on modeling the costs of the active learning protocol (Settles et al., 2009; Haertel et al., 2009) provides some insight on modeling costs associated with interactive learning protocols. A second potentially interesting direction would be to incorporate other semantic resources such as lexical patterns (Hearst, 1992) or Wikipedia-generated gazetteers (Toral and Muñoz, 2006).

Acknowledgments

The authors would like to thank Ming-Wei Chang, Margaret Fleck, Julia Hockenmaier, Alex Klementiev, Ivan Titov, and the anonymous reviewers for their valuable suggestions. This work is supported by DARPA funding under the Bootstrap Learning Program and by MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC.

References

Avrim Blum. 1992. Learning boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386.

Lois Boggess, Rajeev Agarwal, and Ron Davis. 1991. Disambiguation of prepositional phrases in automatically labelled technical text. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 155–159.

Thorsten Brandts and Alex Franz. 2006. Web 1T 5-gram Version 1.

David Cohn, Les Atlas, and Richard Ladner. 1994. Improving generalization with active learning. *Machine Learning*, 15(2):201–222.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1–8.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalization expectation criteria. In *Proc. of International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 595–602.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Robbie Haertel, Kevin D. Seppi, Eric K. Ringger, and James L. Carroll. 2009. Return on investment for active learning. In *NIPS Workshop on Cost Sensitive Learning*.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. the International Conference on Computational Linguistics (COLING)*, pages 539–545.

Yifen Huang and Tom M. Mitchell. 2006. Text clustering with extended user feedback. In *Proc. of International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 413–420.

Xin Li and Dan Roth. 2005. Learning question classifiers: The role of semantic information. *Journal of Natural Language Engineering*, 11(4).

Siau Hong Lim, Li-Lun Wang, and Gerald DeJong. 2007. Explanation-based feature construction. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 931–936.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 337–342.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proc. of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 613–619.

Hema Raghavan and James Allan. 2007. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *Proc. of International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 79–86.

Dan Roth and Kevin Small. 2008. Active learning for pipeline models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 683–688.

Dan Roth and Wen-Tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8.

Burr Settles, Mark Craven, and Lewis Friedland. 2009. Active learning with real annotation costs. In *NIPS Workshop on Cost Sensitive Learning*.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66.

Antonio Toral and Rafael Muñoz. 2006. A proposal to automatically build and maintain gazetteers using wikipedia. In *Proc. of the Annual Meeting of the European Association of Computational Linguistics (EACL)*, pages 56–61.

H. Peyton Young. 1974. An axiomatization of borda's rule. *Journal of Economic Theory*, 9(1):43–52.

Omar F. Zaidan and Jason Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 31–40.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 260–267.

Mining the Web for Reciprocal Relationships

Michael Paul, Roxana Girju, and Chen Li
Linguistics and Computer Science Departments and Beckman Institute,
University of Illinois at Urbana-Champaign
{mjpaul2, girju, chenli}@illinois.edu

Abstract

In this paper we address the problem of identifying reciprocal relationships in English. In particular we introduce an algorithm that semi-automatically discovers patterns encoding reciprocity based on a set of simple but effective pronoun templates. Using a set of most frequently occurring patterns, we extract pairs of reciprocal pattern instances by searching the web. Then we apply two unsupervised clustering procedures to form meaningful clusters of such reciprocal instances. The pattern discovery procedure yields an accuracy of 97%, while the clustering procedures indicate accuracies of 91% and 82%. Moreover, the resulting set of 10,882 reciprocal instances represent a broad-coverage resource.

1 Introduction

Reciprocity is a pervasive concept which has been studied a lot in a wide variety of fields from ethics to game theory where it is analyzed as a highly effective "tit for tat" strategy. The ethic of reciprocity (also known as the *golden rule*), for example, is a moral code born from social interaction: "*Do onto others as you would wish them do onto you*". The golden rule appears in most religions and cultures as a standard used to resolve conflicts.

According to sociologists and philosophers, the concept of reciprocity lies at the foundation of social organization. It strengthens and maintains social relations among people, beyond the basic exchange of useful goods. Thus, the way people conceptualize reciprocity and the way it is expressed in language play an important role in governing people's behavior, judgments, and thus their social interactions.

In this paper we present an analysis of the concept of reciprocity as expressed in English and present a way to model it. In particular we introduce an algorithm that semi-automatically discovers patterns encoding reciprocity based on a set of simple but effective pronoun templates. We then rank the identified patterns according to a scoring function and select the most frequent ones. Using these patterns we query the web and run two unsupervised clustering procedures to form meaningful clusters of reciprocal pattern instances. The pattern discovery procedure yields an accuracy of 97%, while the clustering procedures indicate accuracies of 91% and 82%. Moreover, the resulting set of 10,882 reciprocal instances represent a broad-coverage resource.

Next we define the concept of reciprocity as expressed in English.

Reciprocity in language

The Oxford English Dictionary Online[1] defines reciprocity as "*a state or relationship in which there is mutual action, influence, giving and taking, correspondence, etc., between two parties*", while in WordNet the verb *to reciprocate* means "*to act, feel, or give mutually or in return*".

Reciprocity is defined as a relation between two eventualities e_o (original eventuality) and e_r (reciprocated eventuality), which can occur in various reciprocal constructions. Each eventuality is an event[2] or a state between two participants. Thus, the rela-

[1] http://www.oed.com/

[2] We use the term "event" to denote all those actions or activities performed by people.

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 75–83,
Boulder, Colorado, June 2009.

tion of reciprocity $\Re(e_o(X, Y), e_r(Z, W))$ describes a situation where the eventuality e_r is performed "in return" for e_o. Thus, reciprocity can be seen as a special type of causal relation.

The two arguments of each eventuality represent the subject and the object (direct or indirect), in this order, and they might not all be explicitely stated in the sentence, but can be inferred. Moreover, the participants of the two eventualities might or might not be the same. A few such examples are presented below with the corresponding reciprocity relations:

(1) Mary **argued with** Paul at the station.
$\Re$(argue_with(Mary, Paul), argue_with(Paul Mary)) &
$\Re$(argue_with(Paul, Mary), argue_with(Mary, Paul))

(2) Paul and Mary hate **each other**.
$\Re$(hate(Paul, Mary), hate(Mary, Paul)) &
$\Re$(hate(Mary, Paul), hate(Paul, Mary))

(3) Mary likes Paul **and** he likes her, **too**.
$\Re$(like(Mary, Paul), like(Paul, Mary)) &
$\Re$(like(Paul, Mary), like(Mary, Paul))

(4) Mary likes Paul **for** helping her sister.
$\Re$(help(Paul, Mary's sister), like(Mary,Paul))[3]

As shown in the examples above, in English there are two basic types of reciprocal constructions: mono-clausal reciprocals (involving words such as *(to) hug, to agree/argue with, partner of, mutual(ly), together, each other* – examples (1) and (2)) or sentence-level reciprocals (involving two consecutive clauses – examples (3) and (4)). Most of the sentence-level reciprocals are paraphrased by coordinations or subordinations of two clauses with the same or different predicate and most of the time inverted arguments. They might also manifest various markers as shown in bold in the examples.

In this paper we focus only on sentence-level constructions when the eventualities occur in different consecutive clauses, and when the subject – object arguments of each eventuality are personal pronoun pairs which occur in reverse order in each eventuality. One such example is "**She** *likes* **him** *for helping* **her**". Here the two eventualities are *like(she, he)* and *help(he, she)*. In this example, although the subject of the second verb is not explicitely stated, it is easily inferred. These simplifying assumptions will prove very useful in the semi-supervised pattern discovery procedure to ensure the accuracy of the discovered patterns and their matched instances.

Such a resource of reciprocal event pairs can be very useful in a number of applications, ranging from question answering and textual entailment (since reciprocal event pairs encode a type of causal relation), to behavior analysis of social groups (to monitor cooperation, trustworthiness and personality), and behavior prediction in negotiations.

The paper is organized as follows. In the next section we present relevant previous work. In Section 3 we detail a semi-supervised approach of extracting patterns which encode reciprocity in English. In section 4 we extract pairs of reciprocal instances and cluster them in meaningful clusters. In section 5 we present the experimental data and results. Discussions and conclusion are presented in Section 6.

2 Previous work

Although the concept of reciprocity has been studied a lot in different disciplines such as social sciences (Gergen et al., 1980), anthropology (Sahlins, 1972), economics (Fehr and Gachter, 2000), and philosophy (Becker, 1990), linguists have started to look deeper into this problem only more recently. Moreover, to the best of our knowledge, in computational linguistics the problem is novel.

In linguistics, most of the work on reciprocity focuses on mono-clausal reciprocal constructions, in particular on the quantifiers *each other* and *one another* (Dalrymple et al., 1998; Heim, 1991; König, 2005). Most of this work has been done by language typologists (Maslova and Nedjalkov, 2005; Haspelmath, 2007) who are interested in how reciprocal constructions of these types vary from one language to another and they do this through comparative studies of large sets of world's languages.

In computational linguistics, our pattern discovery procedure extends over previous approaches that use surface patterns as indicators of semantic relations between nouns or verbs ((Hearst, 1998; Chklovski and Pantel, 2004; Etzioni et al., 2004; Turney, 2006; Davidov and Rappoport, 2008) inter alia). We extend over these approaches in two ways: (i) our patterns indicate a new type of relation between verbs, (ii) instead of seed or hook words we

[3]We assume here that the subject of the verb *help* has been recovered and the coreference solved.

use a set of simple but effective pronoun templates which ensure the validity of the patterns extracted.

To the best of our knowledge, the rest of our reciprocity model is novel. In particular, we use a novel procedure which extracts pairs of reciprocal instances and present two novel unsupervised clustering methods which group the instance pairs in meaningful ways. We also present some interesting observations on the data thus obtained and suggest future research directions.

3 Pattern discovery procedure

Our algorithm first discovers clusters of patterns indicating reciprocity in English, and then merges the resulting clusters to identify the final set of reciprocal constructions. In this section we detail the algorithm and evaluate it in subsection 5.2.

3.1 Pronoun templates

In this paper we focus on reciprocal eventualities which occur in two consecutive clauses and have two arguments: a subject and an object. One way to do this is to fully parse each sentence of a corpus and identify coordinations or subordinations of two clauses. Then identify the subject and object arguments of each verb in each clause with the help of a PropBank-style grammatical or semantic role labeler (Kingsbury et al., 2002) and make sure they represent people named entities (as indicated by proper names, personal pronouns, etc.). Since our focus is on reciprocal constructions, we also have to keep in mind that the verbs have to have the same set of arguments (subject-object) in reverse order. Thus, noun and pronoun coreference should also be resolved at this point.

Instead of starting with such a complex and error-prone preprocessing procedure, our algorithm considers a set of pronoun templates, where personal pronouns are anchor words (they have to be matched as such). Each template consists of four personal pronouns corresponding to a subject - object pair in one clause, and a subject - object pair in the other clause. Two such examples are

"[Part1] **I** [Part2] **him** [Part3] **he** [Part4] **me** [Part5]" and

"[Part1] **they** [Part2] **us** [Part3] **we** [Part4] **them** [Part5]",

where [Part1] - [Part5] are partitions identifying any sequence of words. This is an elegant procedure since in English, pronouns have different cases such as nominative and accusative[4] which identify the subject, and respectively the object of an event. This saves us the trouble of parsing a sentence to find the grammatical roles of each verb. In English, there are 30 possible arrangements of nominative - accusative case personal pronoun pairs. Thus we built 30 pronoun templates.

This approach is similar to that of seed words (e.g., (Hearst, 1998)) or hook words (e.g., (Davidov and Rappoport, 2008)) in previous work. However, in our case they are fixed and rich in grammatical information in the sense that they have to correspond to subject - object pairs in consecutive clauses.

Since the first two pronouns in each pronoun template belong to the first clause (C1), and the last two to the second clause (C2), the templates can be restated as [Part1] C1 [Part3] C2 [Part5], with the restriction that partition 3 should not contain any of the four pronouns in the template. C1 denotes "Pronoun1 [Part2] Pronoun2" and C2 denotes "Pronoun3 [Part4] Pronoun4". Partitions 2 and 4 contain the verb phrases (and thus the eventualities) we would like to extract. For speed and memory reasons, we limit their size to no more than 5 words.

Moreover, since the two clauses are consecutive, we hypothesize that they should be very close to each other. Thus, we restrict the size of each partition 1, 3, and 5 to no more than 5 words. We then consider all possible variations of the pattern where the size of each partition varies from 0 to 5. This results in 216 possible combinations (6^3). Moreover, to ensure the accuracy of the procedure, partitions 1 and 5 should be bounded to the left and respectively to the right by punctuation marks, parentheses, or paragraph boundaries. An example of an instance matched by one such pattern is ", **I** *cooked dinner for* **her** *and* **she** *loves* **me** *for that* ."

3.2 Scoring function

One way to compute the prominence of the discovered patterns would be to consider the frequency of each of the five partitions. However, as our preliminary experiments suggest, although individual

[4] In English, the pronouns *you* has the same form in nominative and accusative.

patterns within each partition do often repeat, ranking patterns spanning all three partitions (PART1, PART3, and PART5) is problematic. Patterns with relatively long partitions (more than 2 words each) seldomly occur more than once in the entire corpus. Thus frequency would produce very little differentiation in ranking the patterns.

Thus we developed an alternative scoring system in lieu of frequencies. A sequence of size n ($seq(n)$) is an instance of a pronoun template and a subsequence of size k ($seq(k)$) is simply a substring of the sequence with $k < n$. For example, for the instance "*I love her and she loves me , too*" of length 9, there will be two subsequences of length 8: "*love her and she loves me , too*" and "*I love her and she loves me ,*". Taking into account the frequencies of the subsequences occurring within instances of each partition, we use the following recursive scoring function (n is the length of each subsequence of size n):

$$Score(seq(n)) =$$

$$\begin{cases} Disc(freq(seq(n))) + \\ \sum_{seq(n-1)} Disc(Score(seq(n-1))), & \text{if } n > 1 \\ freq(seq(n)), & \text{if } n = 1 \end{cases} \quad (1)$$

In addition, in order to ensure a valid ranking over the extracted templates with different lengths for each partition, we need to normalize the scores obtained for PART1, PART3, and PART5. In other words, we need to scale the scores obtained for each partition to discount the scores of longer partitions, so that the maximum possible score would remain the same regardless of how long the partition is. So we use the following formula to compute the discount for each of PART1, PART3, and PART5, where n is the length of the subsequence:

$$Disc(Score(seq(n))) =$$

$$\begin{cases} (1.0 - fraction) * \frac{fraction^{m-n}}{m-n+1}, & \text{if } n > 1 \\ \frac{fraction^{m-n}}{m-n+1}, & \text{if } n = 1 \end{cases} \quad (2)$$

Fraction is an empirically predetermined parameter - here set to 0.5. The variable m is the length of the entire PART1, PART3, or PART5 in question.

This allows not only the frequency of the exact pattern to contribute to the score, but also occurrences of similar patterns, although to a lesser extent. And since partitions 1, 3, and 5 constitute the salient parts of the pattern as the environment for the two reciprocal clauses C1 and C2, we take the score to be ranked as $Score(PART1) * Score(PART3) * Score(PART5)$.

We searched the 30 pronoun templates with various partition sizes on a 20 million word English corpus obtained from Project Gutenberg, the largest single collection of free electronic books (over 27,000) (http://www.gutenberg.org) and British National Corpus (BNC), an 100 million word collection of English from spoken and written sources. There were 2,750 instances matched which were ranked by the scoring function. There were 1,613 distinct types of patterns which generated 1,866 distinct pattern instances. Thus, we selected the top 15 patterns, after manual validation. These patterns represent 56% of the data (Table 1). All the other patterns were discarded as having very low frequencies and being very specific.

The manual validation was necessary in order to collapse some of the identified instances into more general classes. For example, the patterns "C1 and C2 to" (e.g., "*He could not hurt me* **and** *I would not wish him to.*"), "C1 and C2 in" (e.g., "*I give you and you take me in.*"), and "C1 and C2 fast said Aunt Jane" (e.g., "*He will come to her* **and** *she can hold him fast said Aunt Jane.*") were collapsed into "C1 and C2". This procedure can be partially solved by identifying complex verbs such as "take in". However, we leave this improvement for future work.

Patterns	**Examples**
C1 [, |; |.] C2	I help him; he helps me.
C1 and C2	He understands her **and** she understands him.
C1 and C2 [right] back	I kissed him **and** and he kissed me **back**.
C1 and C2 for that	They helped us **and** we appreciate them **for that**.
C1 and C2, too	I love her **and** she loves me, **too**.
C1 when C2	He ignores her **when** she scolds him.
C1 whenever C2	He is there for her **whenever** she needs him.
C1 because C2	They tolerate us **because** we helped them.
C1 as much as C2	He loves her **as much as** she loves him.
C1 for C2 (vb-ing)	He thanked her **for** being patient with him.
C1 but C2	I loved her **but** she dumped me.
C1 for what C2	They will punish him **for** what he did to them.
C1 and thus C2	She rejected him **and thus** he killed her.
when C1, C2	**When** he confronted them, they arrested him.
C1 as long as C2	She will stay with him **as long as** he doesn't hurt her.

Table 1: The top 15 reciprocal patterns along with examples.

4 Clustering of Reciprocal Eventualities

It seems reasonable to expect that certain reciprocities could be grouped together. For example, the language used in convincing a person of something could be characterized by verbs such as $e_o = \{convince, promise, assure, beg\}$ and $e_r = \{believe, trust, choose, forgive\}$.

There are many potential uses for this sort of grouping. Having a single group label for multiple reciprocal eventuality pairs would allow us to identify certain language patterns as a particular speech act. Also, such clusters could be useful if one wants to perform a macro-level analysis of reciprocity in a specific domain. For example, examining reciprocal language could be useful in analyzing the nature of a social community or the theme of a literary work. Generalizing over many similar instances, will give us better insight into how people communicate – as reactions (effects) to other people's actions (causes).

Thus, in this section we present a model for clustering the eventualities we extract through the process described in the previous sections. Experimental results are presented in Section 5.

4.1 Representing the data

After obtaining these patterns, we must extract pairs of eventualities of the form (e_o, e_r). This involves both reducing the clauses into a form that is semantically representative of some eventuality, as well as determining the order of the two eventualities (i.e., if they are asymmetric).

As shown in the previous sections, each pattern contains two clauses of the form "$Pronoun_i$ [Part2/4] $Pronoun_j$", where the first pronouns is the subject and the second is the object. From each clause we extract only the non-auxiliary verb, as it carries the most meaning. We first stem the verb and then negate it if it is preceded by *not* or *n't*. For example, "*They do not like him because he snubbed them*" is represented as the eventualities $(e_o, e_r) = (snub, \neg like)$.

Certainly, we are missing important information by excluding phrases and ignoring modality. However, these features can be difficult to capture accurately, and since inaccurate input could degrade the clustering accuracy, in this research we stick with the important and easily-obtainable features.

4.2 Ordering the eventualities

Most patterns entail a particular ordering of the two eventualities, corresponding to symmetric (e.g., "*He loves her* **and** *she loves him*") or asymmetric eventualities (e.g., "*He ignores her* **when** *she scolds him*"). In ambiguous situations (e.g., *He loves her* **and** *she loves him*" and "*He cheated on her* **and** *she still loves him!*"), we determine the order through clues such as the relative temporal ordering of the verbs as determined by their tense (e.g., past or present tense happens before future tense) and whether the verbs denote an action (e.g., "to chase") or a state (e.g., "to love"). For this we rely on our previous work (Girju, 2009) where we identified the order of eventualities based on a set of such features employed in a semi-supervised model whose accuracy is 90.2%.

4.3 Modeling the relationships

The extracted eventuality pairs can be represented as a bipartite graph with a node for all e_o values in one partition, a node for all e_r values in another partition, and an edge between these nodes for each (e_o, e_r) pair. An intuitive way to cluster these eventualities is to find groups of nodes such that each node in one partition has an edge to every node in the other partition and vice versa. This is a form of hard-clustering, as membership in a cluster is strictly *yes* or *no*. The goal is that one could randomly pull an e_o and an e_r from a given cluster and the reciprocity would be valid. For example, "help" and "give" could both be reciprocated by either "thank" or "like". Thus, given a cluster, not only is there a reciprocal relationship between verbs in the e_o group with the verbs in the e_r group, but there is often a kind of similarity relationship between the verbs within each e_o or e_r group.

This approach gives precise and concrete relations between verbs, but while it could be well-suited to some applications (such as knowledge base construction or automatic verb classification (Joanis et al., 2008)[5]) it has disadvantages in the context of grouping these verbs together. The clusters are small and sparse, and the results are difficult to interpret, as there are many overlapping clusters.

[5]These verb classes correspond to some extent to the VerbNet (Kipper et al., 2000) or FrameNet-style (Baker et al., 1998) verb classes such as *admire, judgment*.

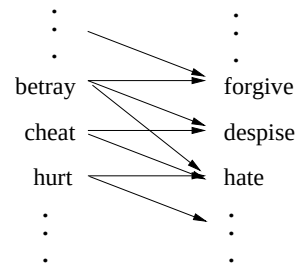


Figure 1: A sample of our data as a bipartite graph. Some edges have been omitted for readability. The nodes {e_o="betray", e_o="cheat", e_r="despise", e_r="hate"} form a cluster with our hard-clustering approach.

We instead adopt a probabilistic framework, which allows us to relax the restrictiveness of the clusters while retaining information about the strength of the pairwise relations. Thus, we design a bimodal mixture model in which we assume that each pair of eventualities (e_o, e_r) belongs to a latent class z, and each class is associated with two distinct multinomial distributions from which the two eventualities are independently drawn. Thus, the probability of generating a particular pair is:

$$P(e_o, e_r) = \sum_{k}^{|Z|} P(e_o|z=k)P(e_r|z=k) \quad (3)$$

Each class can be thought of as a general type of reciprocity, such as an action followed by appreciation, or an attack followed by retaliation. We should be clear that each class is characterized not by a distribution of specific pairs, but by a distribution of e_o verbs and a distribution of e_r verbs. This allows for the classification of (e_o, e_r) pairs that do not appear in the corpus. For example, if we have not seen the pair $(slap, punch)$, but we know that $(slap, hit)$ and $(kick, punch)$ belong to the same class, then it is likely that $(slap, punch)$ is in the same group.

This model can be used in a fully supervised as well as a semi-/unsupervised setting. If some or all of the class labels are unknown, we can learn the model parameters using an estimator such as Expectation-Maximization (EM) (Dempster et al., 1977). For each eventuality pair c_i in a collection C, we update $P(z=k|c_i)$ with the following equation, which represents the E-step:

$$P(z|c_i) \propto P(z)P(e_o^{(c_i)}|z)P(e_r^{(c_i)}|z) \quad (4)$$

In the M-step, we use the following update equations:

$$P(z=k) \propto \alpha + \sum_{i}^{|C|} P(z=k|c_i) \quad (5)$$

$$P(e_o=j|z) = \frac{\beta + \sum_i^{|C|} I(e_o^{(c_i)}=j)P(z|c_i)}{|E_o|\beta + \sum_{j'}\sum_i I(e_o^{(c_i)}=j')P(z|c_i)} \quad (6)$$

where I is a binary indicator function. The equation for $P(e_r = j|z)$ is identical to that for e_o, but with e_r instead[6].

α and β are the hyperparameters of the uniform Dirichlet priors of $P(z)$ and $P(e_*|z)$. They can be tuned to control the level of smoothing; a value of 1.0 is equivalent to the commonly-used Laplace smoothing (Nigam et al., 2000).

4.4 Identifying polarity words

Since we are interested in analyzing how people interact, we would also like to identify the polarity (affective value) associated with each eventuality. Thus, we automatically identify polarity words in both clauses. For this we consider the standard polarity values: *Good, Bad*, and *Neutral*.

In the next section we present in detail the results of the evaluation.

5 Experimental data and results

5.1 Data collection

While the Gutenberg and BNC collections are useful in obtaining the frequent patterns, they do not contain a very large number of eventuality pairs to do meaningful clustering. We thus query the web through Google to easily obtain thousands of examples. We queried each of the top 15 patterns and all pronoun combinations thereof (e.g. "they * us because we * them") and took the top 500 results for each pattern/pronoun combination (15*30*500)[7]. We then extracted the clauses from the result snippets using the procedure outlined in the previous section and ended up with 10,882 pairs

[6] We sometimes use the shorthand $P(z)$ to represent $P(z = k)$, which is updated for each particular value of z.

[7] This is because Google limits traffic. However, in the future we can acquire more instances.

(4,403 unique pairs) since some of the queries had less than 500 matched instances[8].

5.2 Pattern discovery procedure

Since we wanted to see to what extent the 15 most frequently occurring patterns encode reciprocity, we selected a sample of 10 pattern instances matched by each pattern in the text collection obtained from the web. We presented the resulting 130 sentences (a few patterns were not frequent on the web, so we obtained a few less than 10 instances) to 2 judges who evaluated them as encoding reciprocity ('yes') or not ('no'). The judges agreed 97% of the time. Moreover, only 2.3% of the 130 pattern instances did not encode reciprocity as agreed by both judges.

These statistics show that these patterns are highly accurate indicators of reciprocity in English.

5.3 Unsupervised clustering

We can capture pattern instance clusters with no prior labeling by initializing the EM parameters randomly. In our experiments we used $\alpha = 1.0$ and $\beta = 0.01$, with varying numbers of clusters (which we denote as k). EM is sensitive to the initial parameters and can perform poorly due to many local maxima. We thus ran the algorithm several times, and saved the output with the best log-likelihood.

Results from clustering with $k = 6$ are shown in Table 2. The examples shown correspond to a random sample of 10 pairs within the top 10% of $P(e_o, e_r|cluster)$ within each cluster. We find that with larger values of k such as 30 or 50, some of the clusters become noisier, but we can capture finer-grained clusters such as $e_o = \{libel, defame\}$ and $e_r = \{sue, \neg sue\}$.

Upon a close look at the clusters in Table 2, one can see that each one seems to have a central theme. Cluster 1 seems to contain mostly positive actions reciprocated by verbs describing gratitude and appreciation. Cluster 2 has to do with cognition; Cluster 3 has to do with the way people communicate and interact. Cluster 4 captures relationships of need and desire. Cluster 5 is about love and adoration, while Cluster 6 is about hate and other negative events, and how they are reciprocated.

[8]The reciprocity dataset is available for download at http://apfel.ai.uiuc.edu/resources.html.

	Accuracy	
No. instances	6 clusters	9 clusters
Top 20	90.8%	82.2%
20/100	71.7%	66.1%
20/All	34.2%	26.1%

Table 3: Cluster membership accuracy for 6 and 9 clusters.

Cluster membership is defined as $argmax_c$ $P(e_o|c)$ $P(e_r|c)$. We took three samples of pairs: (1) the top 20 pairs with the highest $P(e_o, e_r|c)$ values, (2) a random 20 of the top 10%, and (3) a random 20 of all pairs assigned to each cluster. We presented the pairs to two judges who were asked to identify each pair as belonging to the cluster or not based on coherence; that is, all pairs labeled "yes" appear to be related in some way.

Because we fix the number of clusters, we are making the assumption that each reciprocal pair could be put into one of k groups, which is obviously an assumption that will not hold true. However, if a pair does not fit well into any of the clusters, this should be reflected by a low probability. Thus we can achieve decently high accuracy if we consider only the highest-ranked pairs. The accuracy when considering all pairs is only 34% which means that 34% of reciprocal pairs can be meaningfully placed into only 6 groups, which is actually fairly high.

A big source of inter-annotator disagreement comes from the ambiguity of certain verbs, which is a weakness of our limited representation. For example, without additional information it is not clear how a pair like (know, ask) might relate to others.

5.4 Polarity word identification

For this procedure we used the Subjectivity Clues (Wilson et al., 2005) which provides 8,220 entries. From all the 10,882 eventuality pairs, 40.1% of the total number of words were in the subjectivity lexicon, while 36.9% of the pairs had both words in the subjectivity lexicon.

Table 4 shows all possible combinations of pairs of affective values and their associated probabilities in the corpus. These values are computed for those pairs where both words have known polarity.

As one might expect, each polarity class is most likely to be reciprocated by itself: Good for Good (altruism) and Bad for Bad (retaliation). Furthermore, it is more likely that Good follows Bad ('turn

e_o	e_r	e_o	e_r	e_o	e_r	e_o	e_r	e_o	e_r	e_o	e_r
help	thank	know	respect	call	tell	need	need	love	love	hate	hate
allow	thank	trust	know	ask	give	need	trust	adore	love	attack	hate
invite	thank	tell	trust	tell	help	want	need	understand	love	attack	forgive
rescue	thank	tell	know	tell	tell	want	trust	love	adore	slap	hate
join	thank	know	know	contact	tell	want	want	teach	love	hurt	attack
inform	thank	know	trust	meet	hear	help	need	protect	love	betray	punish
join	admire	know	follow	follow	see	offer	need	feed	love	kill	hate
send	thank	give	let	watch	send	help	help	challenge	love	hit	curse
support	thank	let	like	tell	ignore	help	trust	need	love	treat	dislike
teach	owe	help	marry	confront	tell	love	need	give	love	ruin	shoot

Table 2: The clusters induced after running our unsupervised algorithm with $k = 6$ clusters. The pairs correspond to a sample of the top 10% of pairs with the highest value of $P(e_o, e_r|cluster)$ for each cluster.

	Good	**Bad**	**Neutral**	**Total**
Good	0.90	0.18	0.29	0.63
Bad	0.09	0.82	0.08	0.29
Neutral	0.01	0.002	0.63	0.09

Table 4: All possible combinations of pairs of affective values and their associated probabilities as found in the corpus. The numbers in the table correspond to conditional probabilities P($row_i|col_j$). The Total column indicates the probability of each affective class (P(row_i)).

the other cheek') than that Bad follows Good.

We experimented with incorporating polarity into our clustering process. We defined 9 clusters for each combination of polarity pairs, and initialized the model by labeling the eventuality pairs where the polarity of both words was known. We then ran the EM process on all of the pairs, and since the model parameters were initialized with these 9 groups, their pairs were more likely to fit into clusters that matched their polarity. We found, however, that it had trouble clustering the less-common classes – essentially, everything but (Good, Good) and (Bad, Bad). For example, the cluster that was initialized as (Bad, Good) ended up being dominated by $e_r = thanks$ and mostly positive-polarity words as e_o. This seems to be due to the fact that many of these pairs included $e_r = thanks$ (often in sarcasm, as in "he thanked them for embarrassing him"). But there are many more words associated with *thanks* that are Good, thus those pairs were put into the same group, and the Good verbs eventually overtook the cluster. Problems such as this could perhaps be avoided with more varied labeled data.

We selected a sample of the top 20 pair instances for each of the 9 clusters of polarity pairs and gave them to 2 judges who agreed 82% of the time.

6 Discussion and Conclusions

In this paper we presented an analysis of the concept of reciprocity as expressed in English and a way to model it. The experimental results provided nice insights into the problem, but can be further improved.

We noticed that the identification of polarity words is not always enough to capture the affect of each eventuality. Thus, the text needs to be further processed to identify speech acts corresponding to each clause in the reciprocal patterns. For example, words such as "*sorry*" can be classified as negative, while the entire clause "*I am sorry*" captures the speech act of APOLOGY which is associated with good intentions. As future work, we will recluster the reciprocity pairs.

Another observation concerns the reciprocity property of *magnitude* (cf. (Jackendoff, 2005)) or *equivalence of value* between two eventualities. Most of the time reciprocal eventualities have the same or similar magnitude, as the patterns identified indicate a more or less equivalence of value – i.e., hugs for kisses, thanks for help. And most of these constructions do not focus so much on the magnitude, but on the order in which one eventuality (the effect) is a reaction to the other (the cause). However, a closer look at our data shows that there are also constructions which indicate this property more precisely. One such example is "C1 as much as C2" where even a negation in C1 or C2 might destroy the magnitude balance (e.g., "*She does not love him as much as he loves her.*").

We would like to study this property in more detail as well. This kind of study is very important in the analysis of people's behavior, judgments, and thus their social interactions.

References

C. Baker, Ch. Fillmore, and J. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL 1998)*, pages 86–90, Montreal, Canada.

L. Becker, editor. 1990. *Reciprocity*. University of Chicago Press, Chicago.

T. Chklovski and P. Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP) Conference*.

M. Dalrymple, M. Kazanawa, Y. Kim, S. Mchombo, and S. Peters. 1998. Reciprocal expressions and the concept of reciprocity. *Linguistics and Philosophy*, 21:159–210.

D. Davidov and A. Rappoport. 2008. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automaticaly generated sat analogy questions. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*.

A. P. Dempster, N.M. Laird, and D. B. Rdin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2004. Methods for domain-independent information extraction from the web: An experimental comparison. In *Proceedings of the National Conference on Artificial Intelligence (AAAI) Conference*.

E. Fehr and S. Gachter. 2000. Cooperation and Punishment in Public Goods Experiments. *American Economic Review*, 90:980–994.

K. Gergen, M. Greenberg, and R. Willis, editors. 1980. *Social Exchange: Advances in Theory and Research*. New York: Plenum.

R. Girju. 2009. Reciprocity in language. In *Technical Report*. University of Illinois at Urbana-Champaign.

M. Haspelmath. 2007. Further remarks on reciprocal constructions. In Vladimir P. Nedjalkov, editor, *Reciprocal Constructions*, pages 2087–2115.

M. Hearst. 1998. Automated Discovery of WordNet Relations. In Christiane Fellbaum, editor, *An Electronic Lexical Database and Some of its Applications*, pages 131–151. MIT Press, Cambridge, MA.

I. Heim. 1991. Reciprocity and plurality. *Linguistic Inquiry*, 22:63–101.

R. Jackendoff. 2005. The peculiar logic of value. *Journal of Cognition and Culture*, 6:375–407.

E. Joanis, S. Stevenson, and D. James. 2008. A general feature space for automatic verb classification. *Natural Language Engineering*, 14(3).

P. Kingsbury, M. Palmer, and M. Marcus. 2002. Adding Semantic Annotation to the Penn Treebank. In *Proceedings of the 2nd Human Language Technology Conference (HLT 2002)*, pages 252–256, San Diego, California.

K. Kipper, H. Trang Dang, and M. Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 691–696, Austin, TX.

E. König. 2005. Reciprocity in language: Cultural concepts and patterns of encoding. *Uhlenbeck Lecture*, 23.

E. Maslova and V. Nedjalkov. 2005. Reciprocal constructions. In M. Haspelmath, M. Dryer, D. Gill, and B. Comrie, editors, *The World Atlas of Language Structures*, pages 430–433. New York: Oxford University Press.

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134.

M. Sahlins, editor. 1972. *Stone Age Economics*. Chicago: Aldine-Atherton.

P. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

T. Wilson, J. Wiebe, and P. Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Human Language Technology (HLT/EMNLP) Conference*.

Minimally Supervised Model of Early Language Acquisition

Michael Connor
Department of Computer Science
University of Illinois
connor2@uiuc.edu

Yael Gertner
Department of Psychology
University of Illinois
ygertner@cyrus.psych.uiuc.edu

Cynthia Fisher
Department of Psychology
University of Illinois
cfisher@cyrus.psych.uiuc.edu

Dan Roth
Department of Computer Science
University of Illinois
danr@uiuc.edu

Abstract

Theories of human language acquisition assume that learning to understand sentences is a partially-supervised task (at best). Instead of using 'gold-standard' feedback, we train a simplified "Baby" Semantic Role Labeling system by combining world knowledge and simple grammatical constraints to form a potentially noisy training signal. This combination of knowledge sources is vital for learning; a training signal derived from a single component leads the learner astray. When this largely unsupervised training approach is applied to a corpus of child directed speech, the BabySRL learns shallow structural cues that allow it to mimic striking behaviors found in experiments with children and begin to correctly identify agents in a sentence.

1 Introduction

Sentence comprehension involves assigning semantic roles to sentence constituents, determining who does what to whom. How do young children begin learning to interpret sentences? The structure-mapping view of early verb and syntax acquisition proposes that children treat the number of nouns in the sentence as a cue to its semantic predicate-argument structure (Fisher, 1996), and represent language experience in an abstract format that promotes generalization to new verbs (Gertner et al., 2006).

Theories of human language acquisition assume that learning to understand sentences is naturally a partially-supervised task: the fit of the learner's predicted meaning with the referential context and background knowledge provides corrective feedback (e.g., Pinker (1989)). But this feedback must be noisy; referential scenes provide ambiguous information about the semantic roles of sentence participants. For example, the same participant could be construed as an agent who 'fled' or as a patient who is 'chased'.

In this paper, we address this problem by designing a Semantic Role Labeling system (SRL), equipped with shallow representations of sentence structure motivated by the structure-mapping account, that learns with no gold-standard feedback at all. Instead, the SRL provides its own internally-generated feedback based on a combination of world knowledge and linguistic constraints. As a simple stand-in for world knowledge, we assume that the learner has animacy information for some set of nouns, and uses this knowledge to determine their likely roles. In terms of linguistic constraints, the learner uses simple knowledge about the possible arguments verbs can appear with.

This approach has two goals. The first is to inform theories of language learning by investigating the utility of the proposed internally-generated feedback as one component of the human learner's tools. Second, from an NLP and Machine Learning perspective we propose to inject information into a supervised learning algorithm through a channel other than labeled training data. From both perspectives, our key question is whether the algorithm can use these internally labeled examples to extract general patterns that can be applied to new cases.

By building a model that uses shallow representations of sentences and minimal feedback, but that

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 84–92,
Boulder, Colorado, June 2009.

mimics features of language development in children, we can explore the nature of initial representations of syntactic structure.

1.1 Background

The structure-mapping account of early verb and syntax acquisition makes strong predictions. First, it predicts early use of simple structural cues to sentence interpretation. As soon as children can identify some nouns, they should assign different interpretations to transitive and intransitive sentences, simply by assuming that each noun in the sentence bears a distinct semantic role. Similarly, language-specific syntactic learning should transfer rapidly to new verbs. Second, however, this account predicts striking errors. In "Fred and Ginger danced", an intransitive verb occurs with two nouns. If children interpret any two-noun sentence as if it were transitive, they should mistakenly interpret the order of two nouns in such conjoined-subject intransitive sentences as agent-patient. Experiments with young children support these predictions. 21-month-olds use the number of nouns to understand sentences containing new verbs (Yuan et al., 2007), generalize what they have learned about transitive word-order to new verbs (Gertner et al., 2006), and make the predicted error, treating intransitive sentences containing two nouns as if they were transitive (Gertner and Fisher, 2006). By 25 months, children have learned enough about English syntax to interpret conjoined-subject intransitives differently from transitives (Naigles, 1990).

Our previous computational experiments with a system for automatic semantic role labeling (Connor et al., 2008) suggest that it is possible to learn to assign basic semantic roles based on the simple representations proposed by the structure-mapping view. The classifier's features were limited to lexical information (nouns and verbs only) and the number and order of nouns in the sentence, and trained on a sample of child-directed speech annotated in PropBank (Kingsbury and Palmer, 2002) style. Given this training, our classifier learned to label the first of two nouns as an agent and the second as a patient. Even amid the variability of casual speech, simply representing the target word as the first or the second of two nouns significantly boosts SRL performance (relative to a lexical baseline) on transitive sentences containing novel verbs. This result depends on key assumptions of the structure-mapping view, including abstract representations of semantic roles, and abstract but simple representations of sentence structure. Another approach was taken by (Alishahi and Stevenson, 2007). Their model learned to assign semantic roles without prior knowledge of abstract semantic roles. Instead, it relied on built-in syntactic knowledge and a rich hierarchical representation of semantic knowledge to learn links between sentence structure and meaning.

However, our previous experimental design has a serious drawback that limits its relevance to the study of how children learn their first language. In training, our SRL received gold standard feedback consisting of correctly labeled sentences. Thus when the SRL made a mistake in identifying the semantic role of any noun in a sentence, it received feedback about the 'true' semantic role of this noun. As noted above, this is an unrealistic assumption for the input to human learners.

Here we ask whether an SRL could learn to interpret simple sentences even without gold-standard feedback by relying on world knowledge to generate its own feedback. This internally-generated feedback was based on the following assumptions. First, nouns referring to animate entities are likely to be agents, and nouns referring to inanimate entities are not. Second, each predicate takes at most one agent. Such role uniqueness constraints are typically included in linguistic discussions of thematic roles (Bresnan, 1982; Carlson, 1998). The animacy heuristic is not always correct, of course. For example, in "The door hit you", an inanimate object is the agent of action, and an animate being is the patient. Nevertheless, it is useful for two reasons. First, there is a strong cross-linguistic association between agency and animacy (Aissen, 1999; Dowty, 1991). Second, from the first year of life, children have strong expectations about the capacities of animate and inanimate entities (Baillargeon et al., in press). Given the universal tendency for speakers to talk about animate action on less animate objects, many sentences will present useful training data to the SRL: In ordinary sentences such as "You broke it," feedback generated based on animacy will resemble gold-standard feedback.

2 Learning Model

Our learning task is similar to the full SRL task (Carreras and Màrquez, 2004), except that we classify the roles of individual words rather than full phrases. A full automatic SRL system (e.g. (Punyakanok et al., 2005a)) typically involves multiple stages to 1) parse the input, 2) identify arguments, 3) classify those arguments, and then 4) run inference to make sure the final labeling for the full sentence does not violate any linguistic constraints. Our simplified BabySRL architecture essentially replaces the first two steps with developmentally plausible heuristics. Rather than identifying arguments via a learned classifier with access to a full syntactic parse, the BabySRL treats each noun in the sentence as a candidate argument and assigns a semantic role to it. A simple heuristic collapsed compound or sequential nouns to their final noun, an approximation of the head noun of the noun phrase. For example, 'Mr. Smith' was treated as the single noun 'Smith'. Other complex noun phrases were not simplified in this way. Thus, a phrase such as 'the toy on the floor' would be treated as two separate nouns, 'toy' and 'floor'. This represents the assumption that young children know 'Mr. Smith' is a single name, but they do not know all the predicating terms that may link multiple nouns into a single noun phrase. The simplified learning task of the BabySRL implements a key assumption of the structure-mapping account: that at the start of multiword sentence comprehension children can tell which words in a sentence are nouns (Waxman and Booth, 2001), and treat each noun as a candidate argument.

We further simplify the SRL task such that classification is between two macro-roles: A0 (agent) and A1 (non-agent; all non-A0 arguments). We did so because we reason that this simplified feedback scheme can be primarily informative for a first stage of learning in which learners identify how their language identifies agents vs. non-agents in sentences. In addition, this level of role granularity is more consistent across verbs (Palmer et al., 2005).

For argument classification we use a linear classifier trained with a regularized perceptron update rule (Grove and Roth, 2001). This learning algorithm provides a simple and general linear classifier that works well in other language tasks, and allows us to inspect the weights of key features to determine their importance for classification.

For the final predictions, the classifier uses predicate-level inference to ensure coherent argument assignments. In our task the only active constraints are that all nouns require a tag, and that they have unique labels, which for this restricted case of A0 vs. not A0 means there will be only one agent.

2.1 Training and Feedback

The key feature of our BabySRL lies in the way feedback is provided. Ordinarily, during training, SRL classifiers predict a semantic label for an argument and receive gold-standard feedback about its correct semantic role. Such accurate feedback is not available for the child learner. Children must rely on their own error-prone interpretation of events to supply feedback. This internally-generated feedback signal is presumably derived from multiple information sources, including the plausibility of particular combinations of argument-roles given the current situation (Chapman and Kohn, 1978). Here we model this process by combining background knowledge with linguistic constraints to generate a training signal. The 'unsupervised' feedback is based on: 1) nouns referring to animate entities are assumed to be agents, while nouns referring to inanimate entities are non-agents and 2) each predicate can have at most one agent.

This internally-generated feedback bears some similarities to Inference Based Training (Punyakanok et al., 2005b). In both cases the feedback to local supervised classifiers depends on global constraints. With IBT, feedback for mistakes is only considered after global inference, but for BabySRL the global inference is applied to the feedback itself. Figure 1 gives an overview of the training and testing procedure, making clear the distinction between training and testing inference.

The training data were samples of parental speech to one child ('Sarah'; (Brown, 1973), available via Childes (MacWhinney, 2000)). We trained on parental utterances in samples 1 through 80, recorded at child age 2;3-3;10 years. All verb-containing utterances without symbols indicating long pauses or unintelligible words were automatically parsed with the Charniak parser (Charniak, 1997) and annotated using an existing SRL sys-

tem (Punyakanok et al., 2005a). In this initial pass, sentences with parsing errors that misidentified argument boundaries were excluded. Role labels were hand-corrected using the PropBank annotation scheme. The child-directed speech training set consists of about 8300 tagged arguments over 4700 sentences, of which a majority had a single verb and two labeled nouns[1]. The annotator agreement on this data set ranged between 95-97% at the level of arguments. In the current paper these role-tagged examples provide a comparison point for the utility of animacy-based feedback during training.

Our BabySRL did not receive these hand-corrected semantic roles during training. Instead, for each training example it generated its own feedback based in part on an animacy table. To obtain the animacy table we coded the 100 most frequent nouns in our corpus (which constituted less than 15% of the total number of nouns, but 65% of noun occurrences). We considered 84 of these nouns to be unambiguous in animacy: Personal pronouns and nouns referring to people were coded as animate (30). Nouns referring to objects, body parts, locations, and times, were coded as inanimate (54). The remaining 16 nouns were excluded because they were ambiguous in animacy (e.g., dolls, actions).

We test 3 levels of feedback representing increasing amounts of linguistic knowledge used to generate internal interpretations of the sentences. Using the animacy table, Animacy feedback (**Feedback 1**) was generated as follows: for each noun in training, if it was coded as animate it was labeled A0, if it was coded as inanimate it was labeled A1, otherwise no feedback was given. Because of the frequency of animate nouns this gives a skewed distribution of 4091 animate agents and 1337 inanimate non-agents.

(**Feedback 2**) builds on Feedback 1 by adding another linguistic constraint: if a noun was not found in the animacy-table and there is another noun in the sentence that is labeled A0, then the unknown noun is an A1. In the training set this adds non-agent training examples, yielding 4091 A0 and 2627 A1 examples.

Feedback 1 and Feedback 2 allow two nouns in a sentence to be labeled with A0. **Feedback 3** prevents this; it implements a unique agent constraint that incorporates bootstrapping to make an 'intelligent guess' about which noun is the correct agent. This decision is made based on the current predictions of the classifier. Given a sentence with multiple animate nouns, the classifier predicts a label for each, and the one with the highest score for A0 is declared the true agent and the rest are classified as non-agent. Note that we cannot apply role uniqueness to the A1 (not A0) role, given that this label encompasses multiple non-agent roles. This feedback scheme, allowing at most one agent per sentence, reduces the number of A0 examples and increases the number of A1 examples to 3019 A0 and 3699 A1.

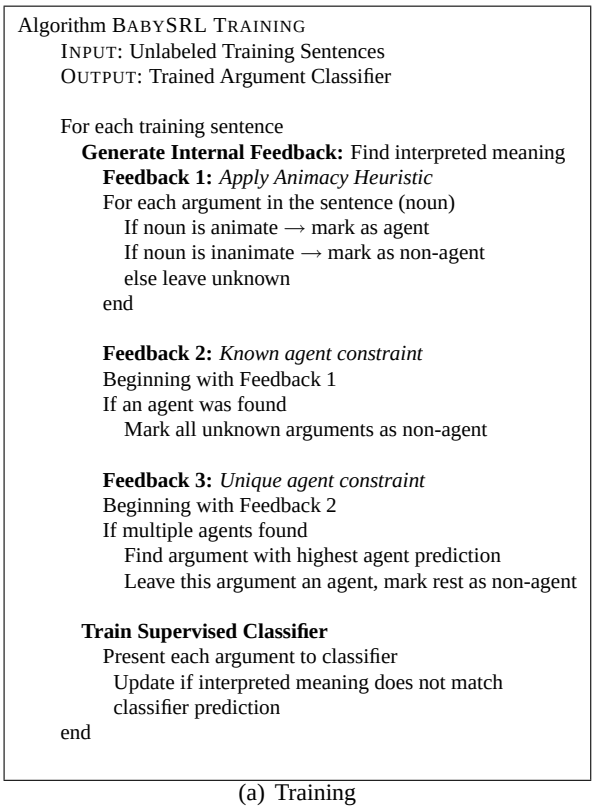
2.2 Feature Sets

The basic feature we propose is the noun pattern feature (NPattern). We hypothesize that children use the number and order of nouns to represent argument structure. The NPattern feature indicates how many nouns there are in the sentence and which noun the target is. For example, in the two-noun sentence 'Did you see it?', 'you' has a feature active indicating that it is the first noun of two. Likewise, for 'it' a feature is active indicating that it is the second of two nouns. This feature is easy to compute once nouns are identified, and does not require fine-grained part-of-speech distinctions.

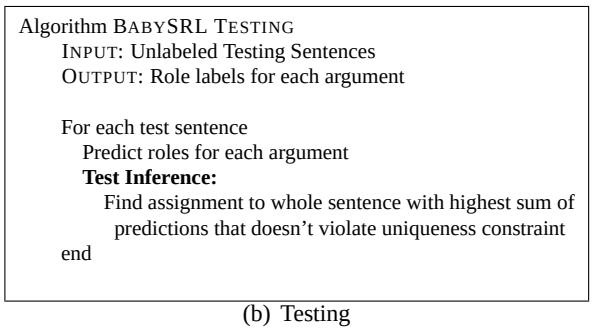
We compare the noun pattern feature to a baseline lexical feature set (Words): the target noun and the root form of the predicate. The NPattern feature set includes lexical features as well as features indicating the number and order of the noun (first of two, second of three, etc.). With gold-standard role feedback, (Connor et al., 2008) found that the NPattern feature allowed the BabySRL to generalize to new verbs: it increased the system's tendency to predict that the first of two nouns was A0 and the second of two nouns A1 for verbs not seen in training.

To the extent that in child-directed speech the first of two nouns tends to be an agent, and agents tend to be animate, we anticipate that with the NPattern feature the BabySRL will learn the same thing, even when provided with internally-generated feedback based on animacy. In Connor et al. (2008) we showed that, because this NPattern feature set represents only the number and order of nouns, with this feature set the BabySRL reproduced the errors chil-

[1]Corpus available at `http://l2r.cs.uiuc.edu/~cogcomp`



(a) Training



(b) Testing

Figure 1: BabySRL training and testing procedures. Internal feedback is generated using animacy plus optional constraints. This feedback is fed to a supervised learning algorithm to create an agent-identification classifier.

dren make as noted in the Introduction, mistakenly assigning agent- and non-agent roles to the first and second nouns in intransitive test sentences containing two nouns. In the present paper, the linguistic constraints provide an additional cause for this error. In addition, as a first step in examining recovery from the predicted error, Connor et al. (2008) added a verb position feature (VPosition) specifying whether the target noun is before or after the verb. Given these features, the BabySRL's classification of transitive and two-noun intransitive test sentences diverged, because the gold-standard training supported the generalization that pre-verbal nouns tend to be agents, and post-verbal nouns tend to be patients. In the present paper we include the VPosition feature for comparison to Connor et al. (2008).

2.3 Testing

To evaluate the BabySRL we tested it with both a held-out sample of child-directed speech, and with constructed sentences containing novel verbs, like those used in the experiments with children described above. These sentences provide a more stringent test of generalization than the customary test on a held-out section of the data. Although the held-out section of data contains unseen sentences, it may contain few unseen verbs. In a held out section of our data, 650 out of 696 test examples contain a verb that was encountered in training. Therefore, the customary test cannot tell us whether the system generalizes what it learned to novel verbs.

All constructed test sentences contained a novel verb ('gorp'). We constructed two test sentence templates: 'A gorps B' and 'A and B gorp', where A and B were replaced with nouns that appeared more than twice in training. For each test sentence template we built a test set of 100 sentences by randomly sampling nouns in two different ways described next.

Full distribution: The first nouns in the test sentences (A) are chosen from the set of all first nouns in our corpus, taking their frequency into account when sampling. The second nouns in the sentences (B) are chosen from the set of nouns appearing as second nouns in the sentence of our corpus. This way of sampling the nouns will maximize the SRL's test performance based on the baseline feature set of lexical information alone (Words). This is so because in our data many sentences have an animate first noun and an inanimate second noun. Based on these words alone the SRL could learn to predict an A0-A1 role sequence for our test sentences. Nevertheless, we expect that when the BabySRL is also given the NPattern feature it should be able to perform better than this high lexical baseline.

Two animate nouns: In these test sentences the A and B nouns are chosen from our list of animate nouns. We chose nouns from this list that were fairly frequent (ranging from 8 to 240 uses in the

corpus), and that occurred roughly equally as the first and second noun. This mimics the sentences used in the experiments with children (e.g., "The girl is kradding the boy!"). The lexical baseline system's tendency to assign an A0-A1 sequence to these nouns should be much lower for these test sentences. We therefore expect the contribution of the NPattern feature to be more apparent in these test sentences.

The test sentences with novel verbs ask whether the classifier transfers its learning about argument role assignment to unseen verbs. Does it assume the first of two nouns in a simple transitive sentence ('A gorps B') is the agent (A0) and the second is not an agent (A1)? In (Connor et al., 2008) we showed that a system with the same feature and representations also over-generalized this rule to two-noun intransitives ('A and B gorp'), mimicking children's behavior. In the present paper this error is over-determined, because the classifier learns only an agent/non-agent contrast, and the linguistic constraints forbid duplicate agents in a sentence. However, for comparison to the earlier paper we test our system on the 'A and B gorp' sentences as well.

3 Experimental Results

Our experiments use internally-generated feedback to train simple, abstract structural features: the NPattern features that proved useful with gold-standard training in Connor et al. (2008). Section 3.1 tests the system on agent-identification in held-out sentences from the corpus, and demonstrates that the animacy-based feedback is useful, yielding SRL performance comparable to that of a system trained with 1000 sentences of gold-standard feedback. Section 3.2 presents the critical novel-verb test data, demonstrating that this system replicates key findings of (Connor et al., 2008) with no gold standard feedback. Using only noisy internally-generated feedback, the BabySRL learned that the first of two nouns is an agent, and generalized this knowledge to sentences with novel verbs.

3.1 Comparing Self Generated Feedback with Gold Standard Feedback

Table 1 reports for the varying feedback schemes, the A0 F1 performance for a system with either lexical baseline feature (Words) or structural features (+NPattern) when tested on a held-out section of the Sarah Childes corpus section 84-90, recorded at child ages 3;11-4;1 years. Agent identification based on lexical features is quite accurate given animacy feedback alone (Feedback 1). As expected, because many agents are animate, the animacy tagging heuristic itself is useful. As linguistic constraints are added via non-A0 inference (Feedback 2), performance increases for both the lexical baseline and NPattern feature-set, because the system experiences more non-A0 training examples.

Feedback	Words	+NPattern
1. Just Animacy	0.72	0.73
2. + non A0 Inference	0.74	0.75
3. + unique A0 bootstrap	0.70	0.74
10 Gold	0.43	0.47
100 Gold	0.61	0.65
1000 Gold	0.75	0.76

Table 1: Agent identification results (A0 F1) on held-out sections of the Sarah Childes corpus. We compare a classifier trained with various amounts of gold labeled data (averaging over 10 different samples at each level of data). For noun pattern features the internally generated bootstrap feedback provides comparable accuracy to training with between 100-1000 fully labeled examples.

When the unique A0 constraint is added (Feedback 3), the lexical baseline performance decreases, because for the first time animate nouns are being tagged as non-agents. With this feedback the NPattern feature set yields a larger improvement over lexical baseline, showing that it extracts more general patterns. We discuss the source of these feedback differences in the novel-verb test section below.

We compared the usefulness of the internally-generated feedback to gold-standard feedback by training a classifier equipped with the same features on labeled sentences. We reduced the SRL labeling for the training sentences to the binary agent/non-agent set, and trained the classifier with 10, 100, or 1000 labeled examples. Surprisingly, the simple feedback derived from 84 nouns labeled with animacy information yields performance equivalent to between 100 and 1000 hand-labeled examples.

	Full Distribution Nouns			Animate Nouns		
Feedback	Words	NPattern	VPosition	Words	NPattern	VPosition
'A gorps B'						
1. Animacy	0.86	0.86	0.87	0.76	0.79	0.70
2. + non A0 Inference	0.87	0.92	0.90	0.63	0.86	0.85
3. + unique A0 bootstrap	0.87	0.95	0.89	0.63	0.82	0.66
'A and B gorp'						
1. Animacy	0.86	0.86	0.84	0.76	0.79	0.68
2. + non A0 Inference	0.87	0.92	0.85	0.63	0.86	0.66
3. + unique A0 bootstrap	0.87	0.95	0.86	0.63	0.82	0.63

Table 2: Percentage of sentences interpreted as agent first (%A0-A1) by the BabySRL when trained on unlabeled data with the 3 internally-generated feedback schemes described in the text. Two different two-noun sentence structures were used ('A gorps B', 'A and B gorp'), along with two different methods of sampling the nouns (Full Distribution, Animate Nouns) to create test sets with 100 sentences each.

3.2 Comparing Structural Features with Lexical Features

The previous section shows that the BabySRL equipped with simple structural features can use internally generated feedback to learn a simple agent/non-agent classification, and apply it to unseen sentences. In this section we probe what the SRL has learned by testing generalization to new verbs in constructed sentences. Table 2 summarizes these experiments. The results are broken down both by what sentence structure is used in test ('A gorps B', 'A and B gorp') and how the nouns 'A' and 'B' are sampled (Full Distribution, Animate Nouns). The results are presented in terms of %A0A1: the percentage of test sentences that are assigned an Agent role for 'A' and a non-Agent role for 'B'.

For the transitive 'A gorps B' sentences, A0A1 is the correct interpretation; A should be the agent. As predicted, when A and B are sampled from the full distribution of nouns, simply basing classification on the Words feature-set already strongly predicts this A0A1 ordering for the majority of cases. This is because the data (language in general, child directed speech in particular here) are naturally distributed such that particular nouns that refer to animates tend to be agents, and tend to appear as first nouns, and those that refer to inanimates tend to be non-agents and second nouns. Thus, a learner representing sentence information in terms of words only succeeds with full-distribution 'A gorps B' test sentences even with the simplest animacy feedback (Feedback 1); the A and B nouns in these test sentences reproduce the learned distribution. Also as predicted, given this simple feedback, the additional higher-level features (NPattern, VPosition) do not improve much upon the lexical baseline. This is due to the strictly lexical nature of the animacy feedback: each lexical item (e.g., 'you' or 'it') will always either be animate or inanimate and therefore either A0 or A1. Therefore, in this case lexical features are the best predictors.

Also as expected, higher-level features (NPattern, and VPosition) improve performance with a more sophisticated self-generated feedback scheme. Adding inferred feedback to label unknown nouns as A1 when the sentence contains a known animate noun (Feedback 2) decreases the ratio of A0 to non-A0 arguments. This feedback is less lexically determined: for nouns whose animacy is unknown, feedback will be provided only if there is another animate noun in the sentence. This leaves room for the abstract structural features to play a role.

Next we test a form of the unique-A0 constraint. In (Feedback 3), in addition to the non-A0 inference added in (Feedback 2), the BabySRL intelligently selects one noun as A0 in sentences with multiple animate nouns. With this feedback we see a striking increase in test performance based on the noun pattern features over the lexical baseline. In principle, this feedback mechanism might permit the classifier to start to learn that animate nouns are not always agents. Early in training, the noun pattern feature learns that first nouns tend to be animate (and therefore interpreted as agents), and it feeds this informa-

tion back into subsequent training examples, generating new feedback that continues to interpret as agents those animate nouns that appear first in sentences containing two animates.

For the nouns sampled from the full distribution we see that structural features improve over the lexical baseline despite the high performance of the lexical baseline. This finding tells us that simple representations of sentence structure can be useful in learning to interpret sentences even with no gold-standard training. Provided only with simple internally-generated feedback based on animacy knowledge and linguistic constraints, the BabySRL learned that the first of two nouns tends to be an agent, and the second of two does not.

The results for the 'A B gorp' test sentences demonstrate an important way in which predictions based on different simple structure representations can diverge. As expected, the NPattern feature makes the same overgeneralization error seen by children and the system in (Connor et al., 2008). However, when the VPosition feature is added, different results are obtained for the 'A gorp B' and 'A and B gorp' sentences. The SRL predicts fewer A0A1 for 'A and B gorp' (it cannot predict the expected A0A0 because of the uniqueness constraint used in test inference).

Next, we replicate our findings by performing the same experiments with test sentences in which both 'A' and 'B' are animate. Because lexical features alone cannot determine if 'A' or 'B' should be the agent, it is a more sensitive test of generalization.

When we look at the lexical baseline for animate sentences, the agent-first percentage is lower compared to the full distribution results, because the word features indicate nearly evenly that both nouns should be agents, so the Words baseline model must rely on small, chance differences in its experience with particular words. This percentage is still well above chance due to the method used to apply inference during testing. Recall that the classifier uses predicate-level inference at test to ensure that only one argument is labeled A0. This inference is implemented using a beam search that looks at arguments in a fixed order and roles from A0 up. Thus in the case of ties there is a preference for first seen solutions, meaning A0A1 in this case. This bias has a large effect on the SRL's baseline performance with the test sentences containing two animate nouns. Despite this high baseline, however, because lexical features alone cannot determine if 'A' or 'B' should be the agent, we are able to see more clearly the improvement gained by including structural features.

Regardless of our testing scheme, we see that as the feedback incorporates more information, both added linguistic constraints and the SRL's own prior learning, the noun pattern structural feature is better used to identify agents beyond the lexical baseline. The largest improvement over this lexical baseline is obtained by combining knowledge of animacy with a single-agent constraint and bootstrapping predictions based on prior learning.

4 Conclusion and Future Work

Conventional approaches to supervised learning require creating large amounts of hand-labeled data. This is labor-intensive, and limits the relevance of the work to the study of how children learn languages. Children do not receive perfect feedback about sentence interpretation. Here we found that our simple SRL classifier can, to a surprising degree, attain performance comparable to training with 1000 sentences of labeled data. This suggests that fully labeled training data can be supplemented by a combination of simple world knowledge (animates make good agents) and linguistic constraints (each verb has only one agent). The combination of these sources provides an informative training signal that allows our BabySRL to learn a high-level semantic task and generalize beyond the training data we provided to it. The SRL learned, based on the distribution of animates in sentences of child-directed speech, that the first of two nouns tends to be an agent. It did so based on representations of sentence structure as simple as the ordered set of nouns in the sentence. This demonstrates that it is possible to learn how to correctly assign semantic roles based on these very simple cues. This together with experimental work (e.g. (Fisher, 1996) suggests that such representations might play a role in children's early sentence comprehension.

Acknowledgments

This research is supported by NSF grant BCS-0620257 and NIH grant R01-HD054448.

References

J. Aissen. 1999. Markedness and subject choice in optimality theory. *Natural Language and Linguistic Theory*, 17:673–711.

A. Alishahi and S. Stevenson. 2007. A computational usage-based model for learning general properties of semantic roles. In *Proceedings of the 2nd European Cognitive Science Conference*.

R. Baillargeon, D. Wu, S. Yuan, J. Li, and Y. Luo. (in press). Young infants expectations about self-propelled objects. In B. Hood and L. Santos, editors, *The origins of object knowledge*. Oxford University Press, Oxford.

J. Bresnan. 1982. *The mental representation of grammatical relations*. MIT Press, Cambridge MA.

R. Brown. 1973. *A First Language*. Harvard University Press, Cambridge, MA.

G. Carlson. 1998. Thematic roles and the individuation of events. In S. D. Rothstein, editor, *Events and Grammar*, pages 35–51. Kluwer, Dordrecht.

X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 shared tasks: Semantic role labeling. In *Proceedings of CoNLL-2004*, pages 89–97. Boston, MA, USA.

R. S. Chapman and L. L. Kohn. 1978. Comprehension strategies in two- and three-year-olds: Animate agents or probable events? *Journal of Speech and Hearing Research*, 21:746–761.

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proc. National Conference on Artificial Intelligence*.

M. Connor, Y. Gertner, C. Fisher, and D. Roth. 2008. Baby srl: Modeling early language acquisition. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, Aug.

D. Dowty. 1991. Thematic proto-roles and argument selection. *Language*, 67:547–619.

C. Fisher. 1996. Structural limits on verb mapping: The role of analogy in children's interpretation of sentences. *Cognitive Psychology*, 31:41–81.

Y. Gertner and C. Fisher. 2006. Predicted errors in early verb learning. In *31st Annual Boston University Conference on Language Development*.

Y. Gertner, C. Fisher, and J. Eisengart. 2006. Learning words and rules: Abstract knowledge of word order in early sentence comprehension. *Psychological Science*, 17:684–691.

A. Grove and D. Roth. 2001. Linear concepts and hidden variables. *Machine Learning*, 42(1/2):123–141.

P. Kingsbury and M. Palmer. 2002. From Treebank to PropBank. In *Proceedings of LREC-2002*, Spain.

B. MacWhinney. 2000. *The CHILDES project: Tools for analyzing talk. Third Edition*. Lawrence Elrbaum Associates, Mahwah, NJ.

L. R. Naigles. 1990. Children use syntax to learn verb meanings. *Journal of Child Language*, 17:357–374.

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

S. Pinker. 1989. *Learnability and Cognition*. Cambridge: MIT Press.

V. Punyakanok, D. Roth, and W. Yih. 2005a. The necessity of syntactic parsing for semantic role labeling. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1117–1123.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2005b. Learning and inference over constrained output. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1124–1129.

S. R. Waxman and A. Booth. 2001. Seeing pink elephants: Fourteen-month-olds's interpretations of novel nouns and adjectives. *Cognitive Psychology*, 43:217–242.

S. Yuan, C. Fisher, Y. Gertner, and J. Snedeker. 2007. Participants are more than physical bodies: 21-month-olds assign relational meaning to novel transitive verbs. In *Biennial Meeting of the Society for Research in Child Development*, Boston, MA.

Learning Where to Look: Modeling Eye Movements in Reading

Mattias Nilsson
Department of Linguistics and Philology
Uppsala University
mattias.nilsson@lingfil.uu.se

Joakim Nivre
Department of Linguistics and Philology
Uppsala University
joakim.nivre@lingfil.uu.se

Abstract

We propose a novel machine learning task that consists in learning to predict which words in a text are fixated by a reader. In a first pilot experiment, we show that it is possible to outperform a majority baseline using a transition-based model with a logistic regression classifier and a very limited set of features. We also show that the model is capable of capturing frequency effects on eye movements observed in human readers.

1 Introduction

Any person engaged in normal skilled reading produces an alternating series of rapid eye movements and brief fixations that forms a rich and detailed behavioral record of the reading process. In the last few decades a great deal of experimental evidence has accumulated to suggest that the eye movements of readers are reflective of ongoing language processing and thus provide a useful source of information for making inferences about the linguistic processes involved in reading (Clifton et al., 2007). In psycholinguistic research, eye movement data is now commonly used to study how experimental manipulations of linguistic stimuli manifest themselves in the eye movement record.

Another related strand of research primarily attempts to understand what determines *when* and *where* the eyes move during reading. This line of research has led to mathematically well specified accounts of eye movement control in reading being instantiated as computational models (Legge et al., 1997; Reichle et al., 1998; Salvucci, 2001; Engbert et al., 2002; McDonald et al., 2005; Feng, 2006; Reilly and Radach, 2006; Yang, 2006). (For a recent overview, see (Reichle, 2006).) These models receive text as input and produce predictions for the location and duration of eye fixations, in approximation to human reading behavior. Although there are substantial differences between the various models, they typically combine both mechanisms of visuomotor control and linguistic processing. Two important points of divergence concern the extent to which language processing influences eye movements and whether readers process information from more than one word at a time (Starr and Rayner, 2001). More generally, the models that have emerged to date are based on different sets of assumptions about the underlying perceptual and cognitive mechanisms that control eye movements. The most influential model so far, the E-Z Reader model (Reichle et al., 1998; Reichle et al., 2003; Pollatsek et al., 2006), rests on the assumptions that cognitive / lexical processing is the engine that drives the eyes through the text and that words are identified serially, one at a time.

Although eye movement models typically have parameters that are fitted to empirical data sets, they are not based on machine learning in the standard sense and their predictions are hardly ever tested on unseen data. Moreover, their predictions are normally averaged over a whole group of readers or words belonging to a given frequency class. In this study, however, we investigate whether saccadic eye movements during reading can be modeled using machine learning. The task we propose is to learn to predict the eye movements of an individual reader reading a specific text, using as training data the eye

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 93–101,
Boulder, Colorado, June 2009.

movements recorded for the same person reading other texts.

Predicting the eye movements of an individual reader on new texts is arguably a hard problem, and we therefore restrict the task to predicting word-based fixations (but not the duration of these fixations) and focus on a first pilot experiment investigating whether we can outperform a reasonable baseline on this task. More precisely, we present experimental results for a transition-based model, using a log-linear classifier, and show that the model significantly outperforms the baseline of always predicting the most frequent saccade. In addition, we show that even this simple model is able to capture frequency effects on eye movements observed in human readers.

We want to emphasize that the motivation for this modeling experiment is not to advance the state of the art in computational modeling of eye movements during reading. For this our model is far too crude and limited in scope. The goal is rather to propose a novel approach to the construction and evaluation of such models, based on machine learning and model assessment on unseen data. In doing this, we want to establish a reasonable baseline for future research by evaluating a simple model with a restricted set of features. In future studies, we intend to investigate how results can be improved by introducing more complex models as well as a richer feature space. More generally, the machine learning approach explored here places emphasis on modeling eye movement behavior with few a priori assumptions about underlying cognitive and physiological mechanisms.

The rest of the paper is structured as follows. Section 2 provides a brief background on basic characteristics of eye movements in reading. The emphasis is on saccadic eye movements rather than on temporal aspects of fixations. Section 3 defines the novel task of learning to predict fixations during reading and discusses different evaluation metrics for this task. Section 4 presents a transition-based model for solving this task, using a log-linear classifier to predict the most probable transition after each fixation. Section 5 presents experimental results for the model using data from the Dundee corpus (Kennedy and Pynte, 2005), and Section 6 contains conclusions and suggestions for future research.

2 Eye Movements in Reading

Perhaps contrary to intuition, the eyes of readers do not move smoothly across a line or page of text. It is a salient fact in reading research that the eyes make a series of very rapid ballistic movements (called saccades) from one location to another. In between saccades, the eyes remain relatively stationary for brief periods of time (fixations). Most fixations last about 200-300 ms but there is considerable variability, both between and within readers. Thus, some fixations last under 100 ms while others last over 500 ms (Rayner, 1998). Much of the variability in fixation durations appears associated to processing ease or difficulty.

The number of characters that is within the region of effective vision on any fixation is known as the perceptual span. For English readers, the perceptual span extends approximately four characters to the left and fifteen characters to the right of the fixation. Although readers fixate most words in a text, many words are also skipped. Approximately 85% of the content words are fixated and 35% of the function words (Carpenter and Just, 1983). Variables known to influence the likelihood of skipping a word are word length, frequency and predictability. Thus, more frequent words in the language are skipped more often than less frequent words. This is true also when word length is controlled for. Similarly, words that occur in constrained contexts (and are thus more predictable) are skipped more often than words in less constrained contexts.

Although the majority of saccades in reading is relatively local, i.e., target nearby words, more distant saccades also occur. Most saccades move the eyes forward approximately 7–9 character spaces. Approximately 15% of the saccades, however, are *regressions*, in which the eyes move back to earlier parts of the text (Rayner, 1998). It has long been established that the length of saccades is influenced by both the length of the fixated word and the word to the right of the fixation (O'Regan, 1979). Regressions often go back one or two words, but occasionally they stretch further back. Such backward movements are often thought to reflect linguistic processing difficulty, e.g., because of syntactic parsing problems. Readers, however, are often unaware of making regressions, especially shorter ones.

3 The Learning Task

We define a *text* T as a sequence of word tokens $(w_1, \ldots, w_n)$, and we define a *fixation sequence* F for T as a sequence of token positions in T $(i_1, \ldots, i_m)$ $(1 < i_k < n)$. The *fixation set* $S(F)$ corresponding to F is the set of token positions that occur in F. For example, the text *Mary had a little lamb* is represented by T = (*Mary*, *had*, *a*, *little*, *lamb*); a reading of this text where the sequence of fixations is *Mary – little – Mary – lamb* is represented by $F = (1, 4, 1, 5)$; and the corresponding fixation set is $S(F) = \{1, 4, 5\}$.

The task we now want to consider is the one of predicting the fixation sequence F for a specific reading event E involving person P reading text T. The training data consist of fixation sequences $F_1, \ldots, F_k$ for reading events distinct from E involving the same person P but different texts $T_1, \ldots, T_k$. The performance of a model M is evaluated by comparing the predicted fixation sequence F_M to the fixation sequence F_O observed in a reading experiment involving P and T. Here are some of the conceivable metrics for this evaluation:

1. **Fixation sequence similarity:** How similar are the sequences F_M and F_O, as measured, for example, by some string similarity metric?

2. **Fixation accuracy:** How large is the agreement between the sets $S(F_M)$ and $S(F_O)$, as measured by 0-1-loss over the entire text, i.e., how large is the proportion of positions that are either in both $S(F_M)$ and $S(F_O)$ (fixated tokens) or in neither (skipped tokens). This can also be broken down into precision and recall for fixated and skipped tokens, respectively.

3. **Fixation distributions:** Does the model predict the correct proportion of fixated and skipped tokens, as measured by the difference between $|S(F_M)|/|T|$ and $|S(F_O)|/|T|$? This can also be broken down by frequency classes of words, to see if the model captures frequency effects reported in the literature.

These evaluation metrics are ordered by an implicational scale from hardest to easiest. Thus, a model that correctly predicts the exact fixation sequence also makes correct predictions with respect to the set of words fixated and the number of words fixated (but not vice versa). In the same fashion, a model that correctly predicts which words are fixated (but not the exact sequence) also correctly predicts the number of words fixated.

In the experiments reported in Section 5, we will use variants of the latter two metrics and compare the performance of our model to the baseline of always predicting the most frequent type of saccade for the reader in question. We will report results both for individual readers and mean scores over all readers in the test set. The evaluation of fixation sequence similarity (the first type of metric) will be left for future work.

4 A Transition-Based Model

When exploring a new task, we first have to decide what kind of model to use. As stated in the introduction, we regard this as a pilot experiment to establish the feasibility of the task and have therefore chosen to start with one of the simplest models possible and see whether we can beat the baseline of always predicting the most frequent saccade. Since the task consists in predicting a sequence of different actions, it is very natural to use a transition-based model, with configurations representing fixation states and transitions representing saccadic movements. Given such a system, we can train a classifier to predict the next transition given the information in the current configuration. In order to derive a complete transition sequence, we start in an initial configuration, representing the reader's state before the first fixation, and repeatedly apply the transition predicted by the classifier until we reach a terminal state, representing the reader's state after having read the entire text. At an abstract level, this is essentially the same idea as in transition-based dependency parsing (Yamada and Matsumoto, 2003; Nivre, 2006; Attardi, 2006). In the following subsections, we discuss the different components of the model in turn, including the transition system, the classifier used, the features used to represent data, and the search algorithm used to derive complete transition sequences.

4.1 Transition System

A transition system is an abstract machine consisting of a set of configurations and transitions between

configurations. A configuration in the current system is a triple $C = (L, R, F)$, where

1. L is a list of tokens representing the left context, including the currently fixated token and all preceding tokens in the text.

2. R is a list of tokens representing the right context, including all tokens following the currently fixated token in the text.

3. F is a list of token positions, representing the fixation sequence so far, including the currently fixated token.

For example, if the text to be read is *Mary had a little lamb*, then the configuration

([*Mary*,*had*,*a*,*little*], [*lamb*], [1,4])

represents the state of a reader fixating the word *little* after first having fixated the word *Mary*.

For any text $T = w_1 \dots w_n$, we define initial and terminal configurations as follows:

1. Initial: $C = ([\,], [w_1, \dots, w_n], [\,])$

2. Terminal: $C = ([w_1, \dots, w_n], [\,], F)$ (for any F)

We then define the following transitions:[1]

1. Progress(n):
   $([\lambda|w_i], [w_{i+1}, \dots, w_{i+n}|\rho], [\phi|i]) \Rightarrow$
   $([\lambda|w_i, w_{i+1}, \dots, w_{i+n}], \rho, [\phi|i, i+n])$

2. Regress(n):
   $([\lambda|w_{i-n}, \dots, w_{i-1}, w_i], \rho, [\phi|i]) \Rightarrow$
   $([\lambda|w_{i-n}], [w_{i-n+1}, \dots, w_i|\rho], [\phi|i, i-n])$

3. Refixate:
   $([\lambda|w_i], \rho, [\phi|i]) \Rightarrow ([\lambda|w_i], \rho, [\phi|i, i])$

The transition Progress(n) models progressive saccades of length n, which means that the next fixated word is n positions forward with respect to the currently fixated word (i.e., $n-1$ words are skipped). In a similar fashion, the transition Regress(n) models regressive saccades of length n. If the parameter n of either Progress(n) or Regress(n) is greater than the number of words remaining in the relevant direction, then the longest possible movement is made instead, in which case Regress(n) leads to a terminal configuration while Progress(n) leads to a configuration that is similar to the initial configuration in that it has an empty L list. The transition Refixate, finally, models refixations, that is, cases where the next word fixated is the same as the current.

To illustrate how this system works, we may consider the transition sequence corresponding to the reading of the text *Mary had a little lamb* used as an example in Section 3:[2]

Init $\Rightarrow$ ([ ], [*Mary*, ..., *lamb*], [ ])
P(1) $\Rightarrow$ ([*Mary*], [*had*, ..., *lamb*], [1])
P(3) $\Rightarrow$ ([*Mary*, ..., *little*], [*lamb*], [1,4])
R(3) $\Rightarrow$ ([*Mary*], [*had*, ..., *lamb*], [1,4,1])
P(4) $\Rightarrow$ ([*Mary*, ..., *lamb*], [ ], [1,4,1,5])

4.2 Learning Transitions

The transition system defined in the previous section specifies the set of possible saccade transitions that can be executed during the reading of a text, but it does not say anything about the probability of different transitions in a given configuration, nor does it guarantee that a terminal configuration will ever be reached. The question is now whether we can learn to predict the most probable transition in such a way that the generated transition sequences model the behavior of a given reader. To do this we need to train a classifier that predicts the next transition for any configuration, using as training data the observed fixation sequences of a given reader. Before that, however, we need to decide on a feature representation for configurations.

Features used in this study are listed in Table 1. We use the notation $L[i]$ to refer to the ith token in the list L and similarly for R and F. The first two features refer to properties of the currently fixated token. Length is simply the character length of the word, while frequency class is an index of the word's frequency of occurrence in representative text. Word frequencies are based on occurrences in the Bristish National Corpus (BNC) and divided into

[1] We use the variables λ, ρ and ϕ for arbitrary sublists of L, R and F, respectively, and we write the L and F lists with their tails to the right, to maintain the natural order of words.

[2] We abbreviate Progress(n) and Regress(n) to P(n) and R(n), respectively.

Feature	Description
CURRENT.LENGTH	The length of the token $L[1]$
CURRENT.FREQUENCYCLASS	The frequency class of the token $L[1]$
NEXT.LENGTH	The length of the token $R[1]$
NEXT.FREQUENCYCLASS	The frequency class of the token $R[1]$
NEXTPLUSONE.LENGTH	The length of the token $R[2]$
NEXTPLUSTWO.LENGTH	The length of the token $R[3]$
DISTANCE.ONETOTWO	The distance, in tokens, between $F[1]$ and $F[2]$
DISTANCE.TWOTOTHREE	The distance, in tokens, between $F[2]$ and $F[3]$

Table 1: Features defined over fixation configurations. The notation $\mathcal{L}[i]$ is used to denote the ith element of list $\mathcal{L}$.

five classes. Frequencies were computed per million words in the ranges 1–10, 11–100, 101–1000, 1001–10000, and more than 10000.

The next four features define features of tokens to the right of the current fixation. For the token immediately to the right, both length and frequency are recorded whereas only length is considered for the two following tokens. The last two features are defined over tokens in the fixation sequence built thus far and record the history of the two most recent saccade actions. The first of these (DISTANCE.ONETOTWO) defines the saccade distance, in number of tokens, that led up to the token currently being fixated. The second (DISTANCE.TWOTOTHREE), defines the next most recent saccade distance, that led up to the previous fixation. For these two features the following holds. If the distance is positive, the saccade is progressive, if the distance is negative, the saccade is regressive, and if the distance amounts to zero, the saccade is a refixation.

The small set of features used in the current model were chosen to reflect experimental evidence on eye movements in reading. Thus, for example, as noted in section 2, it is a well-documented fact that short, frequent and predictable words tend to be skipped. The last two features are included in the hope of capturing some of the dynamics in eye movement behavior, for example, if regressions are more likely to occur after longer progressive saccades, or if the next word is skipped more often if the current word is refixated. Still, it is clear that this is only a tiny subset of the feature space that might be considered, and it remains an important topic for future research to further explore this space and to study the impact of different features.

Given our feature representation, and given some training data derived from reading experiments, it is straightforward to train a classifier for predicting the most probable transition out of any configuration. There are many learning algorithms that could be used for this purpose, but in the pilot experiments we only make use of logistic regression.

4.3 Search Algorithm

Once we have trained a classifier f that predicts the next transition $f(C)$ out of any configuration C, we can simulate the eye movement behavior of a person reading the text $T = (w_1, \ldots, w_n)$ using the following simple search algorithm:

1. Initialize C to $([\,], [w_1, \ldots, w_n], [\,])$.
2. While C is not terminal, apply $f(C)$ to C.
3. Return F of C.

It is worth noting that search will always terminate once a terminal configuration has been reached, even though there is nothing in the transition system that forbids transitions out of terminal configurations. In other words, while the model itself allows regressions and refixations after the last word of the text has been fixated, the search algorithm does not. This seems like a reasonable approximation for this pilot study.

5 Experiments

5.1 Experimental Setup

The experiments we report are based on data from the English section of the Dundee corpus. This sec-

Reader	# sentences	Fixation Accuracy		Fixations			Skips		
		Baseline	Model	Prec	Rec	F1	Prec	Rec	F1
a	136	53.3	70.0	69.9	73.8	71.8	69.0	65.8	67.4
b	156	55.7	66.5	65.2	85.8	74.1	70.3	80.4	75.0
c	151	59.9	70.9	72.5	82.8	77.3	67.4	53.1	59.4
d	162	69.0	78.9	84.7	84.8	84.7	66.0	65.8	65.9
e	182	51.7	71.8	69.1	78.4	73.5	75.3	65.2	69.9
f	157	63.5	67.9	70.9	83.7	76.8	58.7	40.2	47.7
g	129	43.3	56.6	49.9	80.8	61.7	72.2	38.1	49.9
h	143	57.6	66.9	69.4	76.3	72.7	62.8	54.3	58.2
i	196	56.4	69.1	69.6	80.3	74.6	68.2	54.7	60.7
j	166	66.1	76.3	82.2	81.9	82.0	65.0	65.4	65.2
Average	157.8	57.7	69.5	70.3	80.9	75.2	67.5	58.3	62.6

Table 2: Fixation and skipping accuracy on test data; Prec = precision, Rec = recall, F1 = balanced F measure.

tion contains the eye tracking record of ten participants reading editorial texts from The Independent newspaper. The corpus contains 20 texts, each of which were read by all participants. Participants also answered a set of multiple-choice comprehension questions after having finished reading each text. The corpus consists of 2379 sentences, 56212 tokens and 9776 types. The data was recorded using a Dr. Bouis Oculometer Eyetracker, sampling the position of the right eye every millisecond (see Kennedy and Pynte, 2005, for further details).

For the experiments reported here, the corpus was divided into three data sets: texts 1-16 for training (1911 sentences), texts 17-18 for development and validation (237 sentences), and the last two texts 19-20 for testing (231 sentences).

Since we want to learn to predict the observed saccade transition for any fixation configuration, where configurations are represented as feature vectors, it is not possible to use the eye tracking data directly as training and test data. Instead, we simulate the search algorithm on the corpus data of each reader in order to derive, for each sentence, the feature vectors over the configurations and the transitions corresponding to the observed fixation sequence. The instances to be classified then consist of feature representations of configurations while the classes are the possible transitions.

To somewhat simplify the learning task in this first study, we removed all instances of non-local saccades prior to training. Progressions stretching further than five words ahead of the current fixation were removed, as were regressions stretching further back than two words. Refixations were not removed. Thus we reduced the number of prediction classes to eight. Removal of the non-local saccade instances resulted in a 1.72% loss over the total number of instances in the training data for all readers.

We trained one classifier for each reader using logistic regression, as implemented in Weka (Witten and Eibe, 2005) and default options. In addition, we trained majority baseline classifiers for all readers. These models always predict the most frequent saccadic eye movement for a given reader.

The classifiers were evaluated with respect to the accuracy achieved when reading previously unseen text using the search algorithm in 4.3. To ensure that test data were consistent with training data, sentences including any saccade outside of the local range were removed prior to test. This resulted in removal of 18.9% of the total number of sentences in the test data for all readers. Accuracy was measured in three different ways. First, we computed the fixation accuracy, that is, the proportion of words that were correctly fixated or skipped by the model, which we also broke down into precision and recall for fixations and skips separately.[3] Secondly, we compared the predicted fixation distribu-

[3]Fixation/skip precision is the proportion of tokens fixated/skipped by the model that were also fixated/skipped by the reader; fixation/skip recall is the proportion of tokens fixated/skipped by the reader that were also fixated/skipped by the model.

tions to the observed fixation distributions, both over all words and broken down into the same five frequency classes that were used as features (see Section 4). The latter statistics, averaged over all readers, allow us to see whether the model correctly predicts the frequency effect discussed in section 2.

5.2 Results and Discussion

Table 2 shows the fixation accuracy, and precision, recall and F1 for fixations and skips, for each of the ten different models and the average across all models (bottom row). Fixation accuracy is compared to the baseline of always predicting the most frequent saccade type (Progress(2) for readers a and e, and Progress(1) for the rest).

If we consider the fixation accuracy, we see that all models improve substantially on the baseline models. The mean difference between models and baselines is highly significant ($p < .001$, paired t-test). The relative improvement ranges from 4.4 percentage points in the worst case (model of reader f) to 20.1 percentage points in the best case (model of reader e). The highest scoring model, the model of reader d, has an accuracy of 78.9%. The lowest scoring model, the model of reader g, has an accuracy of 56.6%. This is also the reader for whom there is the smallest number of sentences in the test data (129), which means that a large number of sentences were removed prior to testing because of the greater number of non-local saccades made by this reader. Thus, this reader has an unusually varied saccadic behaviour which is particularly hard to model.

Comparing the precision and recall for fixation and skips, we see that while precision tends to be about the same for both categories (with a few notable exceptions), recall is consistently higher for fixations than for skips. We believe that this is due to a tendency of the model to overpredict fixations, especially for low-frequency words. This has a great impact on the F1 measure (unweighted harmonic mean of precision and recall), which is considerably higher for fixations than for skips.

Figure 1 shows the distributions of fixations grouped by reader and model. The models appear reasonably good at adapting to the empirical fixation distribution of individual readers. However, the models typically tend to look at more words than the readers, as noted above. This suggests that the models lack sufficient information to learn to skip words more often. This might be overcome by introducing features that further encourage skipping of words. In addition to word length and word frequency, that are already accounted for, n-gram probability could be included as a measure of predictability, for example.

We also note that there is a strong linear relation between the capability of fitting the empirical distribution well and achieving high fixation accuracy (Pearson's r: -0.91, as measured by taking the differences of each pair of distributions and correlating them with the fixation accuracy of the models).

Figure 2 shows the mean observed and predicted fixation and skipping probability as a function of word frequency class, averaged over all readers. As seen here, model prediction is responsive to frequency class in a fashion comparable to the readers, although the predictions typically tend to exaggerate the observed frequency effect. In the lower to medium classes (1–3), almost every word is fixated. Then there is a clear drop in fixation probability for words in frequency class 4 which fits well with the observed fixation probability. Finally there is another drop in fixation probability for the most frequent words (5). The skipping probabilities for the different classes show the corresponding reverse trend.

6 Conclusion

In this paper we have defined a new machine learning task where the goal is to learn the saccadic eye movement behavior of individual readers in order to predict the sequence of word fixations for novel reading events. We have discussed different evaluation metrics for this task, and we have established a first benchmark by training and evaluating a simple transition-based model using a log-linear classifier to predict the next transition. The evaluation shows that even this simple model, with features limited to a few relevant properties in a small context window, outperforms a majority baseline and captures some of the word frequency effects on eye movements observed in human readers.

This pilot study opens up a number of directions for future research. With respect to modeling, we need to explore more complex models, richer feature spaces, and alternative learning algo-

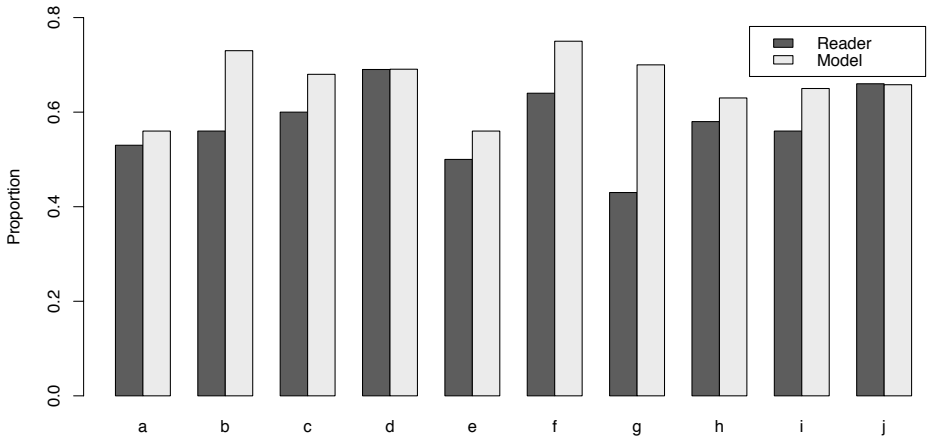


Figure 1: Proportion of fixated tokens grouped by reader and model

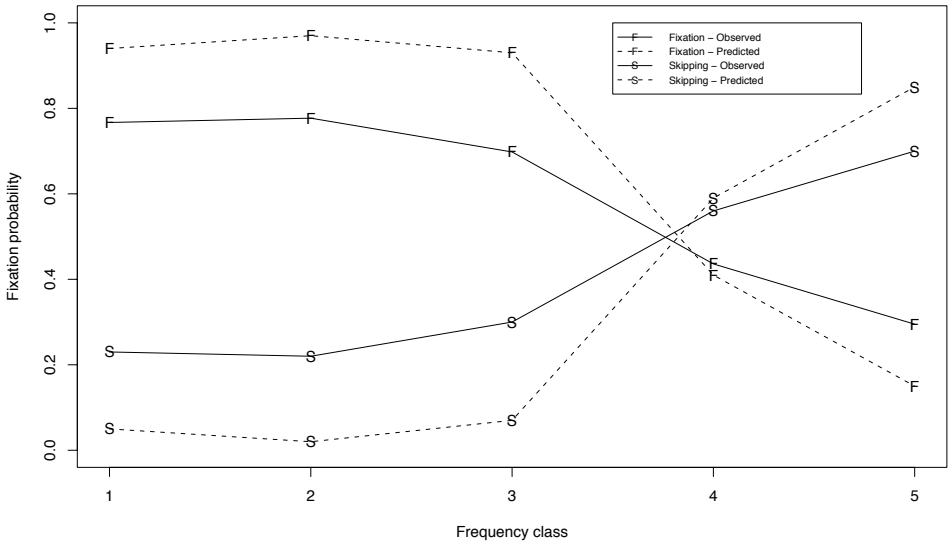


Figure 2: Mean observed and predicted fixation and skipping probability for five frequency classes of words

rithms. For example, given the sequential nature of the task, it seems natural to explore probabilistic sequence models such as HMMs (see for example Feng (2006)). With respect to evaluation, we need to develop metrics that are sensitive to the sequential behavior of models, such as the fixation sequence similarity measure discussed in Section 3, and investigate to what extent results can be generalized across readers. With respect to the task itself, we need to introduce additional aspects of the reading process, in particular the duration of fixations. By pursuing these lines of research, we should be able to gain a better understanding of how machine learning methods in eye movement modeling can inform and advance current theories and models in reading and psycholinguistic research.

References

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 166–170.

Patricia A. Carpenter and Marcel A. Just. 1983. What your eyes do while your mind is reading. In Keith Rayner, editor, *Eye movements in reading: Perceptual and language processes*, pages 275–307. New York: Academic Press.

Charles Clifton, Adrian Staub, and Keith Rayner. 2007. Eye movements in reading words and sentences. In Roger van Gompel, editor, *Eye movements: A window on mind and brain*, pages 341–372. Amsterdam: Elsevier.

Ralf Engbert, André Longtin, and Reinhold Kliegl. 2002. A dynamical model of saccade generation in reading based on spatially distributed lexical processing. *Vision Research*, 42:621–636.

Gary Feng. 2006. Eye movements as time-series random variables: A stochastic model of eye movement control in reading. *Cognitive Systems Research*, 7:70–95.

Alan Kennedy and Joël Pynte. 2005. Parafoveal-on-foveal effects in normal reading. *Vision research*, 45:153–168.

Gordon E. Legge, Timothy S. Klitz, and Bosco S. Tjan. 1997. Mr. Chips: An ideal-observer model of reading. *Psychological Review*, 104:524–553.

Scott A. McDonald, R.H.S. Carpenter, and Richard C. Schillcock. 2005. An anatomically-constrained, stochastic model of eye movement control in reading. *Psychological Review*, 112:814–840.

Joakim Nivre. 2006. *Inductive Dependency Parsing*. Springer.

J. Kevin O'Regan. 1979. Eye guidance in reading: Evidence for the linguistic control hypothesis. *Perception & Psychophysics*, 25:501–509.

Alexander Pollatsek, Erik Reichle, and Keith Rayner. 2006. Tests of the E-Z Reader model: Exploring the interface between cognition and eye movements.

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124:372–422.

Erik Reichle, Alexander Pollatsek, Donald Fisher, and Keith Rayner. 1998. Toward a model of eye movement control in reading. *Psychological Review*, 105:125–157.

Erik Reichle, Keith Rayner, and Alexander Pollatsek. 2003. The E-Z Reader model of eye-movement control in reading: Comparisons to other models. *Behavioral and Brain Sciences*, 26:445–476.

Eric Reichle, editor. 2006. *Cognitive Systems Research*. 7:1–96. Special issue on models of eye-movement control in reading.

Ronan Reilly and Ralph Radach. 2006. Some empirical tests of an interactive activation model of eye movement control in reading. *Cognitive Systems Research*, 7:34–55.

Dario D. Salvucci. 2001. An integrated model of eye movements and visual encoding. *Cognitive Systems Research*, 1:201–220.

Matthew Starr and Keith Rayner. 2001. Eye movements during reading: some current controversies. *Trends in Cognitive Sciences*, 5:156–163.

Ian H. Witten and Frank Eibe. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.

Shun-nan Yang. 2006. A oculomotor-based model of eye movements in reading: The competition/activation model. *Cognitive Systems Research*, 7:56–69.

Monte Carlo inference and maximization for phrase-based translation

Abhishek Arun[*] a.arun@sms.ed.ac.uk
Chris Dyer[†] redpony@umd.edu
Barry Haddow[*] bhaddow@inf.ed.ac.uk
Phil Blunsom[*] pblunsom@inf.ed.ac.uk
Adam Lopez[*] alopez@inf.ed.ac.uk
Philipp Koehn[*] pkoehn@inf.ed.ac.uk

[*]Department of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK

[†]Department of Linguistics
University of Maryland
College Park, MD 20742, USA

Abstract

Recent advances in statistical machine translation have used beam search for approximate NP-complete inference within probabilistic translation models. We present an alternative approach of sampling from the posterior distribution defined by a translation model. We define a novel Gibbs sampler for sampling translations given a source sentence and show that it effectively explores this posterior distribution. In doing so we overcome the limitations of heuristic beam search and obtain theoretically sound solutions to inference problems such as finding the maximum probability translation and minimum expected risk training and decoding.

1 Introduction

Statistical machine translation (SMT) poses the problem: given a foreign sentence f, find the translation e^* that maximises the conditional posterior probability $p(e|f)$. This probabilistic formulation of translation has driven development of state-of-the-art systems which are able to learn from parallel corpora which were generated for other purposes — a direct result of employing a mathematical framework that we can reason about independently of any particular model.

For example, we can train SMT models using maximum likelihood estimation (Brown et al., 1993; Och and Ney, 2000; Marcu and Wong, 2002). Alternatively, we can train to minimise probabilistic conceptions of *risk* (expected loss) with respect to translation metrics, thereby obtaining better results for those metrics (Kumar and Byrne, 2004; Smith and Eisner, 2006; Zens and Ney, 2007). We can also use Bayesian inference techniques to avoid resorting to heuristics that damage the probabilistic interpretation of the models (Zhang et al., 2008; DeNero et al., 2008; Blunsom et al., 2009).

Most models define multiple derivations for each translation; the probability of a translation is thus the sum over all of its derivations. Unfortunately, finding the maximum probability translation is NP-hard for all but the most trivial of models in this setting (Sima'an, 1996). It is thus necessary to resort to approximations for this sum and the search for its maximum e^*.

The most common of these approximations is the max-derivation approximation, which for many models can be computed in polynomial time via dynamic programming (DP). Though effective for some problems, it has many serious drawbacks for probabilistic inference:

1. It typically differs from the true model maximum.

2. It often requires additional approximations in search, leading to further error.

3. It introduces restrictions on models, such as use of only local features.

4. It provides no good solution to compute the normalization factor $Z(f)$ required by many probabilistic algorithms.

In this work, we solve these problems using a Monte Carlo technique with none of the above drawbacks. Our technique is based on a novel Gibbs sampler that draws samples from the posterior distribution of a phrase-based translation model (Koehn et al., 2003) but operates in linear time with respect to the number of input words (Section 2). We show

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 102–110,
Boulder, Colorado, June 2009.

that it is effective for both decoding (Section 3) and minimum risk training (Section 4).

2 A Gibbs sampler for phrase-based translation models

We begin by assuming a phrase-based translation model in which the input sentence, f, is segmented into phrases, which are sequences of adjacent words.[1] Each foreign phrase is translated into the target language, to produce an output sentence e and an alignment a representing the mapping from source to target phrases. Phrases are allowed to be reordered during translation.

The model is defined with a log-linear form, with feature function vector $\mathbf{h}$ and parametrised by weight vector $\boldsymbol{\theta}$, as described in Koehn et al. (2003).

$$P(e, a|f; \boldsymbol{\theta}) = \frac{\exp\left[\boldsymbol{\theta} \cdot \mathbf{h}(e, a, f)\right]}{\sum_{\langle e', a' \rangle} \exp\left[\boldsymbol{\theta} \cdot \mathbf{h}(e', a', f)\right]} \quad (1)$$

The features $\mathbf{h}$ of the model are usually few and are themselves typically probabilistic models indicating e.g, the relative frequency of a target phrase translation given a source phrase (translation model), the fluency of the target phrase (language model) and how phrases reorder with respect to adjacent phrases (reordering model). There is a further parameter Λ that limits how many source language words may intervene between two adjacent target language phrases. For the experiments in this paper, we use $\Lambda = 6$.

2.1 Gibbs sampling

We use Markov chain Monte Carlo (MCMC) as an alternative to DP search (Geman and Geman, 1984; Metropolis and Ulam, 1949). MCMC probabilistically generates sample derivations from the complete search space. The probability of generating each sample is conditioned on the previous sample, forming a Markov chain. After a long enough interval (referred to as the burn-in) this chain returns samples from the desired distribution.

Our MCMC sampler uses Gibbs sampling, which obtains samples from the joint distribution of a set of random variables $X = \{X_1, \ldots, X_n\}$. It starts with some initial state $(X_1 = x_{10}, \ldots, X_n = x_{n0})$, and generates a Markov chain of samples, where each sample is the result of applying a set of *Gibbs operators* to the previous sample. Each operator is defined by specifying a subset of the random variables $Y \subset X$, which the operator updates by sampling from the conditional distribution $P(Y|X \setminus Y)$. The set $X \setminus Y$ is referred to as the Markov blanket and is unchanged by the operator.

In the case of translation, we require a Gibbs sampler that produces a sequence of samples, $\mathcal{S}_1^N = (e_1, a_1) \ldots (e_N, a_N)$, that are drawn from the distribution $P(e, a|f)$. These samples can thus be used to estimate the expectation of a function $h(e, a, f)$ under the distribution as follows:

$$\mathbb{E}_{P(a,e|f)}[h] = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} h(a_i, e_i, f) \quad (2)$$

Taking h to be an indicator function $h = \delta(a, \hat{a})\delta(e, \hat{e})$ provides an estimate of $P(\hat{a}, \hat{e}|f)$, and using $h = \delta(e, \hat{e})$ marginalises over all derivations a', yielding an estimate of $P(\hat{e}|f)$.

2.2 Gibbs operators

Our sampler consists of three operators. Examples of these are depicted in Figure 1.

The RETRANS operator varies the translation of a single source phrase. Segmentation, alignment, and all other translations are held constant.

The MERGE-SPLIT operator varies the source segmentation at a single word boundary. If the boundary is a split point in the current hypothesis, the adjoining phrases can be merged, provided that the corresponding target phrases are adjacent and the phrase table contains a translation of the merged phrase. If the boundary is not a split point, the covering phrase may be split, provided that the phrase table contains a translation of both new phrases. Remaining segmentation points, phrase alignment and phrase translations are held constant.

The REORDER operator varies the target phrase order for a pair of source phrases, provided that the new alignment does not violate reordering limit Λ. Segmentation, phrase translations, and all other alignments are held constant.

To illustrate the RETRANS operator, we will assume a simplified model with two features: a bigram language model P_{lm} and a translation model P_{tm}. Both features are assigned a weight of 1.

[1] These phrases are not necessarily linguistically motivated.

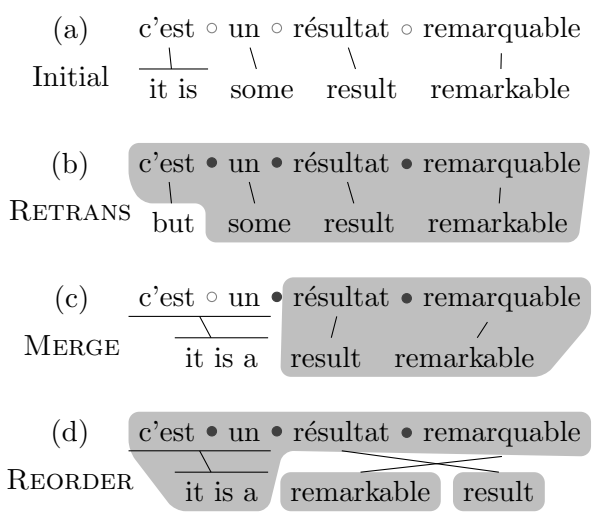


Figure 1: Example evolution of an initial hypothesis via application of several operators, with Markov blanket indicated by shading.

We denote the start of the sentence with S and the language model context with C. Assuming the French phrase *c'est* can be translated either as *it is* or *but*, the RETRANS operator at step (b) stochastically chooses an English phrase, $\hat{e}$ in proportion to the phrases' conditional probabilities.

$$P(\textit{but}|\textit{c'est}, C) = \frac{P_{tm}(\textit{but}|\textit{c'est}) \cdot P_{lm}(\textit{S but some})}{Z}$$

and

$$P(\textit{it is}|\textit{c'est}, C) = \frac{P_{tm}(\textit{it is}|\textit{c'est}) \cdot P_{lm}(\textit{S it is some})}{Z}$$

where

$$\begin{aligned} Z &= P_{tm}(\textit{but}|\textit{c'est}) \cdot P_{lm}(\textit{S but some}) + \\ &\quad P_{tm}(\textit{it is}|\textit{c'est}) \cdot P_{lm}(\textit{S it is some}) \end{aligned}$$

Conditional distributions for the MERGE-SPLIT and REORDER operators can be derived in an analogous fashion.

A complete iteration of the sampler consists of applying each operator at each possible point in the sentence, and a sample is collected after each operator has performed a complete pass.

2.3 Algorithmic complexity

Since both the RETRANS and MERGE-SPLIT operators are applied by iterating over source side word positions, their complexity is linear in the size of the input.

The REORDER operator iterates over the positions in the input and for the source phrase found at that position considers swapping its target phrase with that of every other source phrase, *provided* that the reordering limit is not violated. This means that it can only consider swaps within a fixed-length window, so complexity is linear in sentence length.

2.4 Experimental verification

To verify that our sampler was behaving as expected, we computed the KL divergence between its inferred distribution $\hat{q}(e|f)$ and the true distribution over a single sentence (Figure 2). We computed the true posterior distribution $p(e|f)$ under an Arabic-English phrase-based translation model with parameters trained to maximise expected BLEU (Section 4), summing out the derivations for identical translations and computing the partition term $Z(f)$. As the number of iterations increases, the KL divergence between the distributions approaches zero.

3 Decoding

The task of decoding amounts to finding the single translation e^* that maximises or minimises some criterion given a source sentence f. In this section we consider three common approaches to decoding, maximum translation (MaxTrans), maximum derivation (MaxDeriv), and minimum risk decoding (MinRisk):

$$e^* = \begin{cases} \arg\max_{(e,a)} p(e, a|f) & \text{(MaxDeriv)} \\ \arg\max_e p(e|f) & \text{(MaxTrans)} \\ \arg\min_e \sum_{e'} \ell_{e'}(e) p(e'|f) & \text{(MinRisk)} \end{cases}$$

In the minimum risk decoder, $\ell_{e'}(e)$ is any real-valued loss (error) function that computes the error of one hypothesis e with respect to some reference e'. Our loss is a sentence-level approximation of $(1 - \text{BLEU})$.

As noted in section 2, the Gibbs sampler can be used to provide an estimate of the probability distribution $P(a, e|f)$ and therefore to determine the maximum of this distribution, in other words the most likely derivation. Furthermore, we can marginalise over the alignments to estimate $P(e|f)$

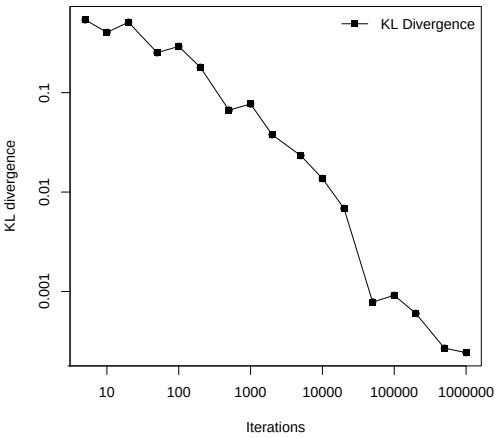


Figure 2: The KL divergence of the true posterior distribution and the distribution estimated by the Gibbs sampler at different numbers of iterations for the Arabic source sentence *r}ys wzrA' mAlyzyA yzwr Alflbyn* (in English, *The prime minister of Malaysia visits the Philippines*).

and so obtain the most likely translation. The Gibbs sampler can therefore be used as a decoder, either running in max-derivation and max-translation mode. Using the Gibbs sampler in this way makes max-translation decoding tractable, and so will help determine whether max-translation offers any benefit over the usual max-derivation. Using the Gibbs sampler as a decoder also allows us to verify that it is producing valid samples from the desired distribution.

3.1 Training data and preparation.

The experiments in this section were performed using the French-English and German-English parallel corpora from the WMT09 shared translation task (Callison-Burch et al., 2009), as well as 300k parallel Arabic-English sentences from the NIST MT evaluation training data.[2] For all language pairs, we constructed a phrase-based translation model as described in Koehn et al. (2003), limiting the phrase length to 5. The target side of the parallel corpus was used to train a 3-gram language model.

[2]The Arabic-English training data consists of the eTIRR corpus (LDC2004E72), the Arabic news corpus (LDC2004T17), the Ummah corpus (LDC2004T18), and the sentences with confidence $c > 0.995$ in the ISI automatically extracted web parallel corpus (LDC2006T02).

For the German and French systems, the DEV2006 set was used for model tuning and the TEST2007 (in-domain) and NEWS-DEV2009B (out-of-domain) sets for testing. For the Arabic system, the MT02 set (10 reference translations) was used for tuning and MT03 (4 reference translations) was used for evaluation. To reduce the size of the phrase table, we used the association-score technique suggested by Johnson et al. (2007a). Translation quality is reported using case-insensitive BLEU (Papineni et al., 2002).

3.2 Translation performance

For the experiments reported in this section, we used feature weights trained with minimum error rate training (MERT; Och, 2003) . Because MERT ignores the denominator in Equation 1, it is invariant with respect to the scale of the weight vector θ — the Moses implementation simply normalises the weight vector it finds by its ℓ_1-norm. However, when we use these weights in a true probabilistic model, the scaling factor affects the behaviour of the model since it determines how peaked or flat the distribution is. If the scaling factor is too small, then the distribution is too flat and the sampler spends too much time exploring unimportant probability regions. If it is too large, then the distribution is too peaked and the sampler may concentrate on a very narrow probability region. We optimised the scaling factor on a 200-sentence portion of the tuning set, finding that a multiplicative factor of 10 worked best for fr-en and a multiplicative factor of 6 for de-en. [3]

The first experiment shows the effect of different initialisations and numbers of sampler iterations on max-derivation decoding performance of the sampler. The Moses decoder (Koehn et al., 2007) was used to generate the starting hypothesis, either in full DP max-derivation mode, or alternatively with restrictions on the features and reordering, or with zero weights to simulate a random initialisation, and the number of iterations varied from 100 to 200,000, with a 100 iteration burn-in in each case. Figure 3 shows the variation of model score with sampler iteration, for the different starting points, and for both language pairs.

[3]We experimented with *annealing*, where the scale factor is gradually increased to sharpen the distribution while sampling. However, we found no improvements with annealing.

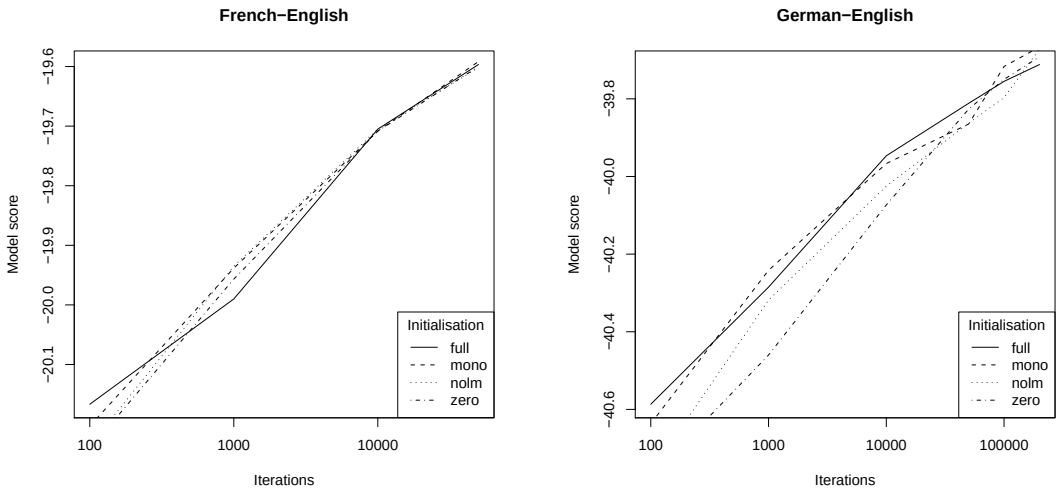


Figure 3: Mean maximum model score, as a function of iteration number and starting point. The starting point can either be the full max-derivation translation (**full**), the monotone translation (**mono**), the monotone translation with no language model (**nolm**) or the monotone translation with all weights set to zero (**zero**).

Comparing the best model scores found by the sampler, with those found by the Moses decoder with its default settings, we found that around 50,000 sampling iterations were required for fr-en and 100,000 for de-en, for the sampler to give equivalent model scores to Moses. From Figure 3 we can see that the starting point did not have an appreciable effect on the model score of the best derivation, except with low numbers of iterations. This indicates that the sampler is able to move fairly quickly towards the maximum of the distribution from any starting point, in other words it has good mobility. Running the sampler for 100,000 iterations took on average 1670 seconds per sentence on the French-English data set and 1552 seconds per sentence on German-English.

A further indication of the dependence of sampler accuracy on the iteration count is provided by Figure 4. In this graph, we show the mean Spearman's rank correlation between the nbest lists of derivations when ranked by (i) model score and (ii) the posterior probability estimated by the sampler. This measure of sampler accuracy also shows a logarithmic dependence on the sample size.

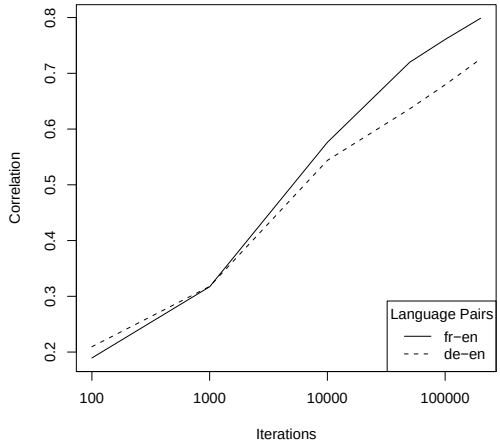


Figure 4: Mean Spearman's rank correlation of 1000-best list of derivations ranked according to (i) model score and (ii) posterior probability estimated by sampler. This was measured on a 200 sentence subset of DEV2006.

3.3 Minimum risk decoding

The sampler also allows us to perform minimum Bayes risk (MBR) decoding, a technique introduced by Kumar and Byrne (2004). In their work, as an approximation of the model probability distribution, the expected loss of the decoder is calculated by summing over an n-best list. With the Gibbs sampler, however, we should be able to obtain a much more accurate view of the model probability distribution. In order to compare max-translation, max-derivation and MBR decoding with the Gibbs sampler, and the Moses baseline, we ran experiments

	fr-en		de-en	
	in	out	in	out
Moses	32.7	19.1	27.4	15.9
MaxD	32.6	19.1	27.0	15.5
MaxT	32.6	19.1	27.4	16.0
MBR	32.6	19.2	27.3	16.0

Table 1: Comparison of the BLEU score of the Moses decoder with the sampler running in max-derivation (MaxD), max-translation (MaxT) and minumum Bayes risk (MBR) modes. The test sets are TEST2007 (in) and NEWS-DEV2009B (out)

on both European language pairs, using both the in-domain and out-of-domain test sets. The sampler was initialised with the output of Moses with the feature weights set to zero and restricted to monotone, and run for 100,000 iterations with a 100 iteration burn-in. The scale factors were set to the same values as in the previous experiment. The relative translation quality (measured according to BLEU) is shown in Table 1.

3.4 Discussion

These results show very little difference between the decoding methods, indicating that the Gibbs sampling decoder can perform as well as a standard DP based max-derivation decoder with these models, and that there is no gain from doing max-translation or MBR decoding. However it should be noted that the model used for these experiments was optimised by MERT, for max-derivation decoding, and so the experiments do not rule out the possibility that max-translation and MBR decoding will offer an advantage on an appropriately optimised model.

4 Minimum risk training

In the previous section, we described how our sampler can be used to search for the best translation under a variety of decoding criteria (max derivation, translation, and minimum risk). However, there appeared to be little benefit to marginalizing over the latent derivations. This is almost certainly a side effect of the MERT training approach that was used to construct the models so as to maximise the performance of the model on its single best derivation, without regard to the shape of the rest of the distribution (Blunsom et al., 2008). In this section we describe a further application of the Gibbs sampler: to do *unbiased* minimum risk training.

While there have been at least two previous attempts to do minimum risk training for MT, both approaches relied on biased k-best approximations (Smith and Eisner, 2006; Zens and Ney, 2007). Since we sample from the whole distribution, we will have a more accurate risk assessment.

The risk, or expected loss, of a probabilistic translation model on a corpus $\mathcal{D}$, defined with respect to a particular loss function $\ell_{\hat{e}}(e)$, where $\hat{e}$ is the reference translation and e is a hypothesis translation

$$\mathcal{L} = \sum_{\langle \hat{e}, f \rangle \in \mathcal{D}} \sum_{e} p(e|f) \ell_{\hat{e}}(e) \tag{3}$$

This value can be trivially computed using equation (2). In this section, we are concerned with finding the parameters $\boldsymbol{\theta}$ that minimise (3). Fortunately, with the log-linear parameterization of $p(e|f)$, $\mathcal{L}$ is differentiable with respect to $\boldsymbol{\theta}$:

$$\frac{\partial \mathcal{L}}{\partial \theta_k} = \sum_{\langle \hat{e}, f \rangle \in \mathcal{D}} \sum_{e} p(e|f) \ell_{\hat{e}}(e) \left(h_k - \mathbb{E}_{p(e|f)}[h_k] \right) \tag{4}$$

Equation (4) is slightly more complicated to compute using the sampler since it requires the feature expectation in order to evaluate the final term. However, this can be done simply by making two passes over the samples, computing the feature expectations on the first pass and the gradient on the second.

We have now shown how to compute our objective (3), the expected loss, and a gradient with respect to the model parameters we want to optimise, (4), so we can use any standard first-order optimization technique. Since the sampler introduces stochasticity into the gradient and objective, we use stochastic gradient descent methods which are more robust to noise than more sophisticated quasi-Newtonian methods like L-BFGS (Schraudolph et al., 2007). For the experiments below, we updated the learning rate after each step proportionally to difference in successive gradients (Schraudolph, 1999).

For the experiments reported in this section, we used sample sizes of 8000 and estimated the gradient on sets of 100 sentences drawn randomly (with replacement) from the development corpus. For a

Training	Decoder	MT03
MERT	Moses Max Derivation	44.6
	Moses MBR	44.8
	Gibbs MBR	44.9
MinRisk	Moses Max Derivation	40.6
	MaxTrans	41.8
	Gibbs MBR	42.9

Table 2: Decoding with minimum risk trained systems, compared with decoding with MERT-trained systems on Arabic to English MT03 data

loss function we use 4-gram $(1 - \text{BLEU})$ computed individually for each sentence[4]. By examining performance on held-out data, we find the model converges typically in fewer than 20 iterations.

4.1 Training experiments

During preliminary experiments with training, we observed on a held-out data set (portions of MT04) that the magnitude of the weights vector increased steadily (effectively sharpening the distribution), but without any obvious change in the objective. Since this resulted in poor generalization we added a regularization term of $||\vec{\theta} - \vec{\mu}||^2/2\sigma^2$ to $\mathcal{L}$. We initially set the means to zero, but after further observing that the translations under all decoding criteria tended to be shorter than the reference (causing a significant drop in performance when evaluated using BLEU), we found that performance could be improved by setting $\mu_{WP} = -0.5$, indicating a preference for a lower weight on this parameter.

Table 2 compares the performance on Arabic to English translation of systems tuned with MERT (maximizing corpus BLEU) with systems tuned to maximise expected sentence-level BLEU. Although the performance of the minimum risk model under all decoding criteria is lower than that of the original MERT model, we note that the positive effect of marginalizing over derivations as well as using minimum risk decoding for obtaining good results on this model. A full exploration of minimum risk training is beyond the scope of this paper, but these initial experiments should help emphasise the versatility of the sampler and its utility in solving a variety of problems. In the conclusion, we will, however, discuss some possible future directions that can be taken to make this style of training more competitive with standard baseline systems.

[4]The ngram precision counts are smoothed by adding 0.01 for n $>$ 1

5 Discussion and future work

We have described an algorithmic technique that solves certain problems, but also verifies the utility of standard approximation techniques. For example, we found that on standard test sets the sampler performs similarly to the DP max-derivation solution and equally well regardless of how it is initialised. From this we conclude that at least for MERT-trained models, the max-derivation approximation is adequate for finding the best translation.

Although the training approach presented in Section 4 has a number of theoretical advantages, its performance in a one-best evaluation falls short when compared with a system tuned for optimal one-best performance using MERT. This contradicts the results of Zens and Ney (2007), who optimise the same objective and report improvements over a MERT baseline. We conjecture that the difference is due to the biased k-best approximation they used. By considering only the most probable derivations, they optimise a smoothed error surface (as one does in minimum risk training), but not one that is indicative of the true risk. If our hypothesis is accurate, then the advantage is accidental and ultimately a liability. Our results are in line with those reported by Smith and Eisner (2006) who find degradation in performance when minimizing risk, but compensate by "sharpening" the model distribution for the final training iterations, effectively maximising one-best performance rather minimising risk over the full distribution defined by their model. In future work, we will explore possibilities for artificially sharpening the distribution during training so as to better anticipate the one-best evaluation conditions typical of MT. However, for applications which truly do require a distribution over translations, such as re-ranking, our method for minimising expected risk would be the objective of choice.

Using sampling for model induction has two further advantages that we intend to explore. First, although MERT performs quite well on models with

small numbers of features (such as those we considered in this paper), in general the algorithm severely limits the number of features that can be used since it does not use gradient-based updates during optimization, instead updating one feature at a time. Our training method (Section 4) does not have this limitation, so it can use many more features.

Finally, for the DP-based max-derivation approximation to be computationally efficient, the features characterizing the steps in the derivation must be either computable independently of each other or with only limited local context (as in the case of the language model or distortion costs). This has led to a situation where entire classes of potentially useful features are not considered because they would be impractical to integrate into a DP based translation system. With the sampler this restriction is mitigated: any function of $h(e, f, a)$ may participate in the translation model subject only to its own computability. Freed from the rusty manacles of dynamic programming, we anticipate development of many useful features.

6 Related work

Our sampler is similar to the decoder of Germann et al. (2001), which starts with an approximate solution and then incrementally improves it via operators such as RETRANS and MERGE-SPLIT. It is also similar to the estimator of Marcu and Wong (2002), who employ the same operators to search the alignment space from a heuristic initialisation. Although the operators are similar, the use is different. These previous efforts employed their operators in a greedy hill-climbing search. In contrast, our operators are applied probabilistically, making them theoretically well-founded for a variety of inference problems.

Our use of Gibbs sampling follows from its increasing use in Bayesian inference problems in NLP (Finkel et al., 2006; Johnson et al., 2007b). Most closely related is the work of DeNero et al. (2008), who derive a Gibbs sampler for phrase-based alignment, using it to infer phrase translation probabilities. The use of Monte Carlo techniques to calculate posteriors is similar to that of Chappelier and Rajman (2000) who use those techniques to find the best parse under models where the derivation and the parse are not isomorphic.

To our knowledge, we are the first to apply Monte Carlo methods to maximum translation and minimum risk translation. Approaches to the former (Blunsom et al., 2008; May and Knight, 2006) rely on dynamic programming techniques which do not scale well without heuristic approximations, while approaches to the latter (Smith and Eisner, 2006; Zens et al., 2007) use biased k-best approximations.

7 Conclusion

We have described a Gibbs sampler for approximating two intractable problems in SMT: maximum translation decoding (and its variant, minimum risk decoding) and minimum risk training. By using Monte Carlo techniques we avoid the biases associated with the more commonly used DP based max-derivation (or k-best derivation) approximation. In doing so we provide a further tool to the translation community that we envision will allow the development and analysis of increasing theoretically well motivated techniques.

Acknowledgments

This research was supported in part by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-001; and by the EuroMatrix project funded by the European Commission (6th Framework Programme). The project made use of the resources provided by the Edinburgh Compute and Data Facility (`http://www.ecdf.ed.ac.uk/`). The ECDF is partially supported by the eDIKT initiative (`http://www.edikt.org.uk/`).

References

P. Blunsom, T. Cohn, and M. Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. of ACL-HLT*.

P. Blunsom, T. Cohn, and M. Osborne. 2009. Bayesian synchronous grammar induction. In *Advances in Neural Information Processing Systems 21*, pages 161–168.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

C. Callison-Burch, P. Koehn, C. Monz, and J. Schroeder, editors. 2009. *Proc. of Workshop on Machine Translations*, Athens.

J.-C. Chappelier and M. Rajman. 2000. Monte-Carlo sampling for NP-hard maximization problems in the

framework of weighted parsing. In *Natural Language Processing – NLP 2000, number 1835 in Lecture Notes in Artificial Intelligence*, pages 106–117. Springer.

J. DeNero, A. Bouchard, and D. Klein. 2008. Sampling alignment structure under a Bayesian translation model. In *Proc. of EMNLP*.

J. R. Finkel, C. D. Manning, and A. Y. Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proc. of EMNLP*.

S. Geman and D. Geman. 1984. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

U. Germann, M. Jahr, K. Knight, D. Marcu, and K. Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *Proceedings of ACL*. Association for Computational Linguistics, July.

J. Johnson, J. Martin, G. Foster, and R. Kuhn. 2007a. Improving translation quality by discarding most of the phrasetable. In *Proc. of EMNLP-CoNLL*, Prague.

M. Johnson, T. Griffiths, and S. Goldwater. 2007b. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proc. of NAACL-HLT*, pages 139–146, Rochester, New York, April.

P. Koehn, F. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*, pages 48–54, Morristown, NJ, USA.

P. Koehn, H. Hoang, A. B. Mayne, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL Demonstration Session*, pages 177–180, June.

S. Kumar and W. Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Processings of HLT-NAACL*.

D. Marcu and W. Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proc. of EMNLP*, pages 133–139.

J. May and K. Knight. 2006. A better n-best list: Practical determinization of weighted finite tree automata. In *Proc. of NAACL-HLT*.

N. Metropolis and S. Ulam. 1949. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341.

F. Och and H. Ney. 2000. A comparison of alignment models for statistical machine translation. In *Proc. of COLING*, Saarbrucken, Germany, July.

F. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL*, pages 160–167, Sapporo, Japan, July.

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

N. N. Schraudolph, J. Yu, and S. Günter. 2007. A stochastic quasi-Newton method for online convex optimization. In *Proc. of Artificial Intelligence and Statistics*.

N. N. Schraudolph. 1999. Local gain adaptation in stochastic gradient descent. Technical Report IDSIA-09-99, IDSIA.

K. Sima'an. 1996. Computational complexity of probabilistic disambiguation by means of tree grammars. In *Proc. of COLING*, Copenhagen.

D. A. Smith and J. Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proc. of COLING-ACL*, pages 787–794.

R. Zens and H. Ney. 2007. Efficient phrase-table representation for machine translation with applications to online MT and speech translation. In *Proc. of NAACL-HLT*, Rochester, New York.

R. Zens, S. Hasan, and H. Ney. 2007. A systematic comparison of training criteria for statistical machine translation. In *Proc. of EMNLP*, pages 524–532, Prague, Czech Republic.

H. Zhang, C. Quirk, R. C. Moore, and D. Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *Proc. of ACL: HLT*, pages 97–105, Columbus, Ohio.

Investigating Automatic Alignment Methods for Slide Generation from Academic Papers

Brandon Beamer and Roxana Girju
Department of Linguistics
University of Illinois
Urbana, IL
{bbeamer,girju}@illinos.edu

Abstract

In this paper we investigate the task of automatic generation of slide presentations from academic papers, focusing initially on slide to paper alignment. We compare and evaluate four different alignment systems which utilize various combinations of methods used widely in other alignment and question answering approaches, such as TF-IDF term weighting and query expansion. Our best aligner achieves an accuracy of 75% and our findings show that for this application, average TF-IDF scoring performs more poorly than a simpler method based on the number of matched terms, and query expansion degrades aligner performance.

1 Introduction

Automatic generation of slide presentations is a task the Computational Linguistics community has not yet pursued in much depth. A robust system capable of generating slide presentations from papers would save the author much tedium when organizing her presentations. In this paper we investigate this task from a novel perspective. While others have developed interesting approaches to slide generation from documents by modeling the problem in a unique way (Utiyama and Hasida, 1999; Shibata and Kurohashi, 2005), the aim of the research this paper initiates is to discover how humans create slide presentations, focusing more specifically on academic papers. Thus we take a corpus-based approach to the problem, and as a first step focus on the task of automatically aligning slide presentations to academic papers.

We built a corpus of 296 slide-paper pairs and implemented four slide to paper aligners which utilize popular information retrieval methods such as TF-IDF term weighting and query expansion. In this paper we show that, in this application, TF-IDF term weighting is inferior to a simpler scoring mechanism based only on the number of matched terms and query expansion degrades aligner performance. Our best aligner achieves an accuracy of 75%.

2 Related Work

Automatic slide generation from documents is a thus far under-investigated topic. Utiyama and Hasida (1999) generate slides from GDA[1] (global document annotation) tagged documents. They detect topics within the documents by analyzing GDA coreference links, modeled each slide as a topic and itemized elaborations (which were also tagged with the GDA tag set). Shibata and Kurohashi (2005) convert Japanese documents to slide representation by parsing their discourse structures and representing the resulting tree in an outline format. While (Utiyama and Hasida, 1999) and (Shibata and Kurohashi, 2005) generate slides from documents by modeling the task in creative ways, we aim to learn something deeper regarding how humans actually go about the task. Creating a corpus of slide/paper pairs will enable us to study the intricacies involved in how real humans approach this task.

Our current focus is slide to paper (region) alignment, which can be categorized best as alignment between monolingual comparable corpora, but

[1]The GDA tag set is designed to allow machines to automatically infer the underlying structure of documents. More information is available at http://i-content.org/gda.

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 111–119,
Boulder, Colorado, June 2009.

could also be easily construed as document passage retrieval, which is a well-researched topic in the Information Retrieval community. Barzilay and Elhadad (2003) incorporate context to facilitate alignment between monolingual comparable corpora by first learning paragraph matching rules in a supervised way, and then refining the alignment at the sentence level within paragraphs. Nelken and Shieber (2008) used TF-IDF term weighting with logistic regression to align sentences from pericopes in the gospels of the new testament. Callan (1994) analyzed various ways to define document passages and identified three main passage types, discourse (based on physical structure of the document), semantic (based on topic boundaries), and window (based on token distance) and suggests that while discourse passages may be an attractive way to define and retrieve document passages, due to reasons related to sloppy writing, visual aids, or other factors, paragraph boundaries may not be the best indicators of content boundaries. Our alignment task differs from that of (Barzilay and Elhadad, 2003) and (Nelken and Shieber, 2008) in two ways. First, Barzilay and Elhadad (2003) and Nelken and Shieber (2008) align like-chunks between the two documents. That is, they are either aligning sentences to sentences or paragraphs to paragraphs. In our task we are aligning slide regions which are usually bullets spanning at most a couple lines, to paper regions which can be a whole paragraph long. Second, Barzilay and Elhadad (2003) and (Nelken and Shieber, 2008) are working with comparable corpora in which the same information is assumed to be present in each document, but expressed in a different way. We are not able to necessarily make this assumption, in fact we show in this paper that as much as half of the information in slide presentations may not be present in the corresponding paper.

The concept of query expansion that we implement in some of our aligners is also not new. Voorhees (1994) suggests that query expansion tends to help performance with short, incomplete queries but degrades performance with longer, more complete queries. van der Plas and Tiedemann (2008) investigated several types of lexico-semantic information for query expansion in their question answering system. They found that expansions that bridge the terminology gap (synonyms, etc.) did not result in improvement but expansions that bridge the knowledge gap (words belonging to the same subject field) did. In this paper, to get an idea of the baseline performance of query expansion with regard to our unique task, we implement a more rudimentary form of query expansion which only expands synonyms of terms. Since our slide regions don't vary much in length, it's hard to say how our results relate to the findings of Voorhees (1994). Our results partially support (van der Plas and Tiedemann, 2008) in that our implementation only bridges the terminology gap, and isn't very successful.

3 The Corpus

The first step to understanding how humans generate slides from papers is to collect real-world examples of academic papers and corresponding slide presentations. To build our corpus, we searched the internet for web pages containing workshop proceedings from various fields using generic queries such as 'workshop slide paper'. The collected papers and presentations come from a variety of fields but tend to be focused generally on science and technology. Workshop proceedings are an ideal source for our data because they often provide the papers and slide presentations side-by-side. Using this strategy, we manually extracted 296 slide-paper pairs. The papers were downloaded in PDF format and the slides were a mixture of PDF and Powerpoint formats. Before working with these files, we converted them to a custom XML format which represents relevant parts of the original data as logical regions. In the case of slides, *regions* include bullets, headings, and other text spans. In the case of papers, *regions* include regions (or passages) which correspond to paragraphs, section headings, and list items.

To work with PDF data, we convert it to a custom XML format which represents logical chunks or *regions* of the paper. In our approach we delimit regions by orthographic boundaries. Orthographic boundaries delimit the physical structure of a paper and describe the paper in a physical fashion in terms of paragraphs, headings, bullets, etc. We do recognize that there are other ways to define paper regions though. As Callan (1994) observes, academic papers could also be represented via semantic boundaries which delimit the topical structure of papers and de-

scribe them in terms of where new topics are introduced and where old ones are no longer discussed. We prefer using orthographic boundaries in our approach for two reasons. First, detecting orthographic boundaries can be accomplished with simple heuristics while topic boundary detection requires more sophisticated methods[2], thus implementation is easier. Second, because orthographic boundaries are far less subjective than topic boundaries, it's easier to verify the validity of orthographic boundaries than semantic ones.

Preprocessing Powerpoint files is significantly simpler than PDF files. To convert the Powerpoint data to our custom XML, we first convert the Powerpoint file to an OpenOffice.org[3] ODP file via the *document converter* tool that comes standard with OpenOffice. ODP files are already encoded with a rich XML which already describes physical regions such as list items, bullets, and other text, so region identification is unnecessary. We only needed to implement a filter that translates the available data to the custom XML format.

4 Alignment Methods

Discovering how humans generate slide presentations from papers starts with observing where slide regions originate from. We make the general assumption that a slide region either a) is a summarization (excerpt or abstract) from the associated paper, or b) comes from other sources including but not limited to the author's personal (world and/or specific) knowledge. A complete alignment module would thus need to be able to discern if the information in a region comes from the target paper or if it does not. When it does, the task of the aligner is then to choose the region in the paper that is summarized or from which the excerpt is taken. Our original hypothesis was that the vast majority of the data in a given slide presentation would come from the target paper and concluded that a reasonable first attempt at building an aligner could be made under this assumption.

We approach the task of aligning slide regions to paper regions with methods popular in information retrieval. When aligning a slide region to a paper region, we treat the slide region as a search query and the target regions as documents in the information retrieval sense. We compare two TF-IDF based scoring methods and the effect of query expansion by building four different aligners, each of which corresponds to one combination of scoring type and usage of query expansion. Table 1 shows a diagram indicating which aligners have which features.

Aligner	Scoring	Query Expansion
A	Method 1	No
B	Method 1	Yes
C	Method 2	No
D	Method 2	Yes

Table 1: Features implemented by each aligner.

To prepare both the slide region and paper for alignment, certain preprocessing tasks are executed by all our aligners. The general procedure all our aligners follow is outlined below:

1. For each token in each region in the paper, the token's TF-IDF score is calculated, where the token's term frequency is the frequency of the token's stem in the region and the term's document frequency is the number of regions containing the token's stem.
2. The slide region is tokenized and part-of-speech tagged with the SNoW tagger (Roth, 1998) and non-content words are removed. We consider content words to be any token which is either a noun, adjective, verb, adverb, or cardinal number.
3. Each token in the slide region is stemmed and, in the case of aligners B and D, query expansion is performed.
4. A score is calculated for each region in the target paper according to the scoring function implemented by the aligner–method 1 for aligners A and B and method 2 for aligners C and D.

These methods are presented in detail below.

4.1 Scoring Methods

In this paper we investigate two scoring methods, which we'll refer to as scoring method 1 and scoring method 2. Scoring method 1 is implemented by aligners A and B and is equivalent to the average TF-IDF score of the search terms relative to the target region. I.e. to calculate the score for a slide region relative to a target paper region with method 1, the TF-IDF scores of all the search terms are added and the sum is divided by the number of terms, and the

[2]Reynar (1998) provides a detailed overview of the basic topic detection and segmentation methods

[3]OpenOffice.org is a freely available office suite available at `http://www.openoffice.org`.

target region with the highest average score wins. Scoring method 2 is implemented by aligners C and D and is based on the quantity of matched terms, reverting to scoring method 1 only in the case of a tie. Thus, to calculate the score for a slide region relative to a target paper region with method 2, the number of search terms with non-zero TF-IDF scores for the paper region is counted and the region with the largest number of such search terms wins. In the case of a tie, the average score is calculated as it is in method 1 and the region with the highest average score wins the tie.

With either scoring method, a zero score results in the system predicting that the slide region is not derived from any paper region.

4.2 Query Expansion

One common problem with rudimentary TF-IDF based information retrieval systems is that matching tokens must have a form identical to the search terms. Hence, synonyms and other semantically-related words that probably should match do not. Query expansion is one way to consider terms which are semantically near, but orthographically different from the search terms. The general principle of query expansion is that, via an external knowledge base, semantic neighbors of search terms are added to the search query before the score is calculated.

Our implementation of query expansion is utilized by aligners B and D and uses Wordnet (Fellbaum, 1998) to extract synonyms of search terms. When a slide region undergoes query expansion our aligner executes the following steps:

1. The search terms are part-of-speech tagged using the SNoW part-of-speech tagger (Roth, 1998) and lemmatized with a morphological analyzer[4].
2. The resulting lemmas and parts of speech are used to query Wordnet for matching synsets.
3. Synonyms for all retrieved synsets are recorded.
4. When scoring occurs, the TF-IDF score of a search term changes from the score of the stem to the maximum score among the stem and all its synonyms. In the case of scoring method 2, a search term matches if it stem is found in the target region or if any of its synonyms' stems are found.

[4]The morphological analyzer we use is called *morpha* and is freely available and can be downloaded at http://www.informatics.susx.ac.uk/research/groups/nlp/carroll/morph.html

5 Evaluation

To evaluate our aligners, we manually checked the alignment of each on four randomly chosen slide presentation-paper pairs. We refer to these presentations here as *P1*, *P2*, *P3*, and *P4*. Collectively, these four presentations with their respective papers amount to 587 alignment decisions which were evaluated according to the following guidelines. If the slide region is either an excerpt from the chosen paper region or if the slide region is an abstract of the chosen paper region, the alignment is judged as good. In cases where the matching excerpt or abstract text spans more than one paper region, the alignment is judged as good if the aligner selected any of the involved regions. Otherwise, the alignment is judged as bad and an error code is recorded. The three error codes we utilize are *BR*, *NR*, and *ER*. *BR* is short for "better region" and indicates that the alignment is bad because the chosen paper region is not the paper region from which the slide region is extracted or generated, but such a region does indeed exist. *NR* is short for "no region" and indicates that the alignment is bad because there is no region in the paper to which the slide region should be aligned. *ER* is short for "existing region" and indicates that the alignment is bad because the aligner decided there was no paper region to which the slide region should be aligned, but in fact there was. Also, the type of each slide region was recorded as either *frontmatter* (which covers text spans such as titles, authors, dates, and addresses), *outline*, *heading*, *bullet*, or *diagram*. Table 2 illustrates the composition of the four presentations insofar as slide region type is concerned.

The distribution of slide region types is not surprising. Table 2 shows that two of our presentations included diagrams and the other two did not, and that bullets not surprisingly account for more slide regions than any other region type.

5.1 Alignability of Slide Regions

Table 3 shows the percentage of slide regions which have a target paper region (i.e. the percentage of alignable slide regions). One surprising observation is that only about half (57%) of the slide bullets were alignable. This goes against our initial hypothesis that the vast majority of slide regions would come

Presentation	Frontmatter	Outline	Heading	Bullet	Diagram
P1	3/174 (1.7%)	0/174 (0.0%)	5/174 (2.9%)	74/174 (42.5%)	92/174 (52.9%)
P2	9/181 (5.0%)	9/181 (5.0%)	34/181 (18.8%)	129/181 (71.3%)	0/181 (0.0%)
P3	5/114 (4.4%)	1/114 (0.9%)	52/114 (45.6%)	55/114 (48.2%)	0/114 (0.0%)
P4	5/118 (4.2%)	1/118 (0.8%)	13/118 (11.0%)	47/118 (39.8%)	52/118 (44.0%)
Total	22/587 (3.7%)	11/587 (1.9%)	104/587 (17.7%)	305/587 (52.0%)	144/587 (24.5%)

Table 2: Breakdown of slide text spans by type. Columns correspond to slide text span types. Percentages in each column measure the fraction of text spans which are of the given type.

from the associated paper, and not from the author's knowledge.

Another important observation from the data in table 3 is that the fraction of slide regions which are alignable for any given presentation can vary wildly. 82% of P4's regions were alignable while 60% of P3's and only 14% of P1's regions were alignable.

5.2 Aligner Accuracy

Tables 4 and 5 show the *raw accuracy* and *alignable accuracy* of the four aligners respectively. Raw accuracy is the number of slide regions correctly aligned out of the total number of slide regions. Alignable accuracy is the percentage of alignable slide regions which were aligned correctly.

Given the surprising results that a large percentage of slide regions need not come from the paper, any fully fledged slide to paper aligner would need a module which first filters out the unalignable slide regions. Because such a module is not implemented in our aligners, as our aligners make the assumption that each slide region has a corresponding paper region, we limit most of our accuracy evaluation to *alignable* accuracy rather than raw accuracy.

From tables 4 and 5 we can easily see the importance of such a filtering module. As our best aligner, which achieves an average alignable accuracy of 75%, only achieves an average raw accuracy of 50%.

5.3 Error Analysis

Tables 6 and 7 show what percentage of an aligner's errors correspond to which error types. Because our aligners are based on term matching, the only way for them to predict no alignment is for the average TF-IDF score of the terms to be zero (no matching terms anywhere). Because this is a very rare event, ER-type errors are also extremely rare, and are excluded from our error analysis.

We can see from tables 6 and 7 that our poorer aligners (A and B) have a fairly even split between BR-type and NR-type errors, while our better aligners (C and D) have a far greater percentage of NR-type errors, indicating that the features we are investigating can only reduce BR-type errors. This verifies the importance of the proposed alignability module which first filters out unalignable slide regions.

5.4 Error Reduction

Tables 8 and 9 analyze how well query expansion and scoring method 2 reduce errors by measuring the percentage of errors made by one aligner, which were not made by another. Four pairings of aligners are considered: A and B, A and C, B and D, and C and D. By comparing aligner A to B and C to D, we have one measure of the error reduction achieved by adding query expansion to an aligner. If the addition of query expansion enables an aligner to correctly align slide regions which its query expansion-less counterpart could not, then we should see large percentages of errors being corrected when comparing aligner A to B and C to D. By comparing aligner A to C and B to D, we have a measure of the error reduction achieved by implementing scoring method 2 instead of method 1.

Tables 8 and 9 show that aligner D significantly reduced aligner B's errors and aligner C significantly reduced aligner A's errors, but aligner B did not improve much on A, nor did D on C. In other words, adding query expansion did not significantly reduce errors, but using scoring method 2 instead of 1 did.

6 Discussion

6.1 On Alignability

Before mentioning alignment performance, it is important to notice from our data that there is great variety among slide presentations. For example, ta-

Presentation	Frontmatter	Outline	Heading	Bullet	Diagram	Overall
P1	3/3 (100.0%)	0/0	0/5 (0.0%)	21/74 (28.4%)	0/92 (0.0%)	24/174 (13.8%)
P2	9/9 (100.0%)	8/9 (88.9%)	24/34 (70.6%)	104/129 (80.6%)	0/0	145/181 (80.1%)
P3	5/5 (100.0%)	0/1 (0.0%)	48/52 (92.3%)	15/55 (27.3%)	0/0	68/114 (59.5%)
P4	4/5 (80.0%)	0/1 (0.0%)	11/13 (74.6%)	33/47 (70.2%)	49/52 (94.2%)	97/118 (82.2%)
Total	21/22 (95.5%)	8/11 (72.7%)	83/104 (79.8%)	173/305 (56.7%)	49/144 (34.0%)	334/587 (56.9%)

Table 3: Breakdown of alignable slide text spans by type. Columns correspond to slide text span types. Percentages in each column measure the fraction of text spans of that type which are alignable. E.g. of the 129 bullets in presentation P2, 104 are alignable. The "Overall" column measures the fraction of all text spans which are alignable. E.g. of the 181 text spans in presentation P2, 145 are alignable.

Presentation	Aligner A	Aligner B	Aligner C	Aligner D
P1	34/174 (19.5%)	129/174 (16.7%)	37/174 (21.3%)	35/174 (20.1%)
P2	71/181 (39.2%)	64/181 (35.4%)	101/181 (55.8%)	97/181 (53.6%)
P3	66/114 (57.9%)	64/114 (56.1%)	77/114 (67.5%)	77/114 (67.5%)
P4	50/118 (42.4%)	48/118 (40.7%)	78/118 (66.1%)	77/118 (65.3%)
Total	221/587 (37.6%)	205/587 (34.9%)	293/587 (49.9%)	286/587 (48.7%)

Table 4: Raw accuracy. Each column corresponds to one of the four aligners evaluated. Percentages measure the fraction of text spans which were aligned correctly.

Presentation	Aligner A	Aligner B	Aligner C	Aligner D
P1	12/24 (50.0%)	9/24 (37.5%)	15/24 (62.5%)	15/24 (62.5%)
P2	63/145 (43.4%)	56/145 (38.6%)	93/145 (64.1%)	90/145 (62.1%)
P3	55/68 (80.9%)	54/68 (79.4%)	66/68 (97.1%)	67/68 (98.5%)
P4	49/97 (50.5%)	47/97 (48.5%)	77/97 (79.4%)	76/97 (78.4%)
Total	179/334 (53.6%)	166/334 (49.7%)	251/334 (75.1%)	248/334 (74.3%)

Table 5: Alignable accuracy. Each column corresponds to one of the four aligners evaluated. Percentages measure the fraction of alignable text spans which were aligned correctly.

	Aligner A		Aligner B	
Presentation	BR	NR	BR	NR
P1	11/140 (7.9%)	128/140 (91.4%)	14/145 (9.7%)	130/145 (89.7%)
P2	82/110 (74.5%)	28/110 (25.5%)	89/117 (76.1%)	28/117 (23.9%)
P3	13/48 (27.1%)	35/48 (72.9%)	14/50 (28.0%)	36/50 (72.0%)
P4	48/68 (70.6%)	20/68 (29.4%)	50/70 (71.4%)	20/70 (28.6%)
Total	154/366 (42.1%)	211/366 (57.7%)	167/382 (43.7%)	214/382 (56.0%)

Table 6: Error type breakdown for aligners A and B. Columns correspond to specific types of alignment errors. "BR" is short for "better region" and "NR" is short for "no region". An error of type "BR" means that the aligner choose an incorrect region in the paper, and a better region existed. An error of type "NR" means the aligner choose an incorrect region, and there was no correct region.

	Aligner C		Aligner D	
Presentation	BR	NR	BR	NR
P1	8/137 (5.8%)	128/137 (93.4%)	8/139 (5.8%)	130/139 (93.5%)
P2	52/80 (65.0%)	28/80 (35.0%)	55/84 (65.5%)	29/84 (34.5%)
P3	2/37 (5.4%)	35/37 (94.6%)	1/37 (2.7%)	36/37 (97.3%)
P4	20/40 (50.0%)	20/40 (50.0%)	21/41 (51.2%)	20/41 (48.8%)
Total	82/294 (27.9%)	211/294 (71.8%)	85/301 (28.2%)	215/301 (71.4%)

Table 7: Error type breakdown for aligners C and D. Columns correspond to specific types of alignment errors. "BR" is short for "better region" and "NR" is short for "no region". An error of type "BR" means that the aligner choose an incorrect region in the paper, and a better region existed. An error of type "NR" means the aligner choose an incorrect region, and there was no correct region.

	Aligner A → B			Aligner A → C		
Presentation	BR	NR	Overall	BR	NR	Overall
P1	0/11 (0.0%)	0/128 (0.0%)	0/140 (0.0%)	4/11 (36.4%)	0/128 (0.0%)	4/140 (2.9%)
P2	0/82 (0.0%)	0/28 (0.0%)	0/110 (0.0%)	38/82 (46.3%)	0/28 (0.0%)	38/110 (34.5%)
P3	0/13 (0.0%)	0/35 (0.0%)	0/48 (0.0%)	11/13 (84.6%)	0/35 (0.0%)	11/48 (22.9%)
P4	0/48 (0.0%)	0/20 (0.0%)	0/68 (0.0%)	31/48 (64.6%)	0/20 (0.0%)	31/68 (45.6%)
Total	0/154 (0.0%)	0/211 (0.0%)	0/366 (0.0%)	84/154 (54.5%)	0/211 (0.0%)	84/366 (23.0%)

Table 8: Error reduction between aligners A and B, and between aligners A and C. Major columns correspond to aligner pairs and minor columns correspond to error types. A pair denoted by $X \rightarrow Y$ indicates that the corresponding percentages are measuring the fraction of slide text spans aligned incorrectly by aligner X, which were aligned correctly by aligner Y. E.g. from this table you can see that in presentation P1, aligner A incorrectly aligned 140 text spans. 11 of them were BR-type errors and 128 of them were NR-type errors. Four of aligner A's BR-type errors were aligned correctly by aligner C.

	Aligner B → D			Aligner C → D		
Presentation	BR	NR	Overall	BR	NR	Overall
P1	7/14 (50.0%)	0/130 (0.0%)	7/145 (4.8%)	0/8 (0.0%)	0/128 (0.0%)	0/137 (0.0%)
P2	42/89 (47.2%)	0/28 (0.0%)	42/117 (35.9%)	1/52 (1.9%)	0/28 (0.0%)	1/80 (1.2%)
P3	13/14 (92.9%)	0/36 (0.0%)	13/50 (26.0%)	1/2 (50.0%)	0/35 (0.0%)	1/37 (2.7%)
P4	32/50 (64.0%)	0/20 (0.0%)	32/70 (45.7%)	1/20 (5.0%)	0/20 (0.0%)	1/40 (2.5%)
Total	94/167 (56.3%)	0/214 (0.0%)	94/382 (24.6%)	3/82 (3.7%)	0/211 (0.0%)	3/294 (1.0%)

Table 9: Error reduction between aligners B and D, and between aligners C and D. Major columns correspond to aligner pairs and minor columns correspond to error types. A pair denoted by $X \rightarrow Y$ indicates that the corresponding percentages are measuring the fraction of slide text spans aligned incorrectly by aligner X, which were aligned correctly by aligner Y. E.g. from this table you can see that in presentation P1, aligner B incorrectly aligned 145 text spans. 14 of them were BR-type errors and 130 of them were NR-type errors. 7 of aligners B's BR-type errors were correctly aligned by aligner D.

ble 3 shows that 28% of P1's bullets were alignable, while 81% of P2's were alignable. P1 and P4 both contained diagrams, but only P4's diagram existed in the paper. Our initial hypothesis was that the vast majority of slide regions would either be excerpts or abstracts from/of the paper regions. Table 3 shows that a nontrivial amount of slide regions does not map to the paper at all. Also, tables 6 and 7 show that as a result, NR-type errors make up the majority of the errors made by the better aligners. Thus, the data indicates that the task of slide-presentation generation is highly dependent on the end purpose the presentation will serve, as well as the target audience and other factors. We will focus more on identifying these factors in future research. Once identified, these factors should be quantified and controlled in future corpora of presentation-paper pairs used for this task.

6.2 On Scoring Methods and Query Expansion

Our results clearly show that, for this task, query expansion has little or negative impact on aligners and that scoring method 2 is indeed superior to scoring method 1. Tables 4 and 5 show that aligner C consistently outperforms aligner A and aligner D consistently outperforms aligner B, especially when limited to alignable slide regions. Hence, scoring method 2 is better than method 1. We can also see from tables 4 and 5 that aligner B consistently under-performs A and aligner D consistently underperforms C, which shows that query expansion does not improve performance and in fact, it degrades it. Tables 8 and 9 show the same results from a different perspective: aligner C correctly aligned 55% of the aligner A's erroneous alignable slide regions and aligner D correctly aligned 56% of aligner B's erroneous alignable slide regions. But aligner B did not catch any of aligner A's errors and aligner D only caught 4% of aligner C's errors – but ended up making more in the end anyway.

With regard to query expansion, there are two possibilities. Query expansion was not very helpful here because either (a) slide authors tend to use wording identical to that in the paper, or (b) using synonyms from Wordnet is not aggressive enough and we should consider expanding our query expansion approach to include hypernyms, immediate hyponyms, and other semantically related terms. We think the data suggests that (a) is more the case than (b). If (b) were the case, including synonyms in our search should have improved the performance, just not by a lot. In actually, aligner B performed worse

on average than aligner A, and likewise with aligner D when compared to C. Synonyms are semantically closer to the original term than hypernyms, hyponyms, or other semantically related terms, and our results show that introducing this small amount of semantic distance is (a little bit) detrimental. By adding hypernyms and other relations, only a wider, less focused group of terms will be introduced which will probably just result in more false positives.

One possible criticism against our argument for (a) could be that our implementation of query expansion performed poorly because we don't word sense disambiguate, and thus we introduce synonyms from incorrect senses of each term. This probably isn't the case because the search terms are not in isolation, but are part of a larger query. For an incorrect paper region to be select based on an error of this type, it would have to contain many of the terms in the query as well as the semantically inaccurate sense of the one in question. This situation is unlikely due to one of the most basic assumptions made when sense disambiguating: that context restricts the possible senses of any word. So, if a paper region contains many of the terms in a slide region, it is unlikely that it will also contain the off-topic, semantically awkward term pertaining to a bad sense of one of them.

With regard to scoring methods. Average TF-IDF scoring is probably ineffective in this application because of the nature of paper regions. When retrieving whole documents given a search query, one document's contents are probably independent of any other, so terms related to the document's topic are stated explicitly. Paper regions, however, are in the context of each other. The topic of one can be very similar to another, only because it's nearby, not because of the terms explicitly mentioned in the region. Add to this the fact that paper regions are extremely non-uniform in length and TF-IDF scores end up skewed.

6.3 On Improvement

There is a lot of room for improvement on slide to paper alignment. As mentioned previously in section 6.1, unalignable slide regions account for a much larger portion of the slide presentations than our initial hypothesis predicted; around 70% of the errors made by our better aligners (C and D) were NR-type errors, meaning the alignment was bad because the system selected a paper region when in fact there was no correct paper region. A robust slide to paper aligner would need to have a module capable of filtering out unalignable slide regions. If this task were solved and implemented on our better aligners, raw accuracy would raise from 50% to about 75% on average which is nearing the level of robustness necessary for real-world applications.

We also suggest that, in regard to alignable slide regions, performance would be significantly boosted by taking context into account, both on the slide and paper side. We noticed during evaluation that many of the BR-type errors occurred when the slide region in question lacked the necessary terms, but the terms existed in nearby slide regions. Examples of this include when for instance, the title is broken across two lines and the second line only has a word or two in it, or when a heading is rather non-descriptive but the sub-bullets beneath it contain many relevant terms to the topic. Incorporating terms of nearby slide regions (perhaps in query-expansion fashion), rather than just treating each one as an independent search query will certainly boost performance.

Likewise on the paper end, it is reasonable to assume that in most cases, the topic of one region is similar to the topics of adjacent regions. And just as terms from nearby slide regions could supplement term-poor slide regions, terms from nearby paper regions could supplement term-poor paper regions.

7 Conclusion

In this paper we investigated the task of automatic slide to paper alignment. We built a corpus of slide-paper pairs and used four presentations from it to evaluate four aligners which utilize methods such as TF-IDF term weighting and query expansion. We showed that query expansion does not improve performance in our application and that TF-IDF term weighting is inferior to a much simpler scoring mechanism based on the number of matched terms. For future improvements, we suggest that a module capable of robustly filtering out unalignable slide regions is necessary. We also suggest that performance can be improved by taking context into account and using terms in nearby regions to supplement both slide regions and paper regions.

References

Regina Barzilay and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

James P. Callan. 1994. Passage-level evidence in document retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Christiane Fellbaum. 1998. *WordNet - An Electronic Lexical Database*. Cambridge MA: MIT Press.

Rani Nelken and Stuart M. Shieber. 2008. Towards robust context-sensitive sentence alignment for monolingua corpora. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*.

Jeffrey C. Reynar. 1998. *Topic Segmentation: Algorithms and Applications*. Ph.D. thesis, University of Pennsylvania.

Dan Roth. 1998. Learning to resolve natural lanuage ambiguities: A unified approach. In *Proceedings of the 15th Conference of the American Association for Artificial Intelligence (AAAI)*.

Tomohide Shibata and Sadao Kurohashi. 2005. Automatic slide generation based on discourse structure analysis. In *Proceedings of the second international joint conference on natural language processing (IJCNLP)*.

Masao Utiyama and Koiti Hasida. 1999. Automatic slide presentation from semantically annotated documents. In *Proceedings of the workshop held in conjunction with the 37th annual meeting of the Association for Computational Linguistics (ACL)*.

Lonneke van der Plas and Jörg Tiedemann. 2008. Using lexico-semantic information for query expansion in passage retrieval for question answering. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*.

Ellen M. Voorhees. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Glen, Glenda or Glendale: Unsupervised and Semi-supervised Learning of English Noun Gender

Shane Bergsma
Department of Computing Science
University of Alberta
Edmonton, Alberta
Canada, T6G 2E8
bergsma@cs.ualberta.ca

Dekang Lin
Google, Inc.
1600 Amphitheatre Parkway
Mountain View
California, 94301
lindek@google.com

Randy Goebel
Department of Computing Science
University of Alberta
Edmonton, Alberta
Canada, T6G 2E8
goebel@cs.ualberta.ca

Abstract

English pronouns like *he* and *they* reliably reflect the gender and number of the entities to which they refer. Pronoun resolution systems can use this fact to filter noun candidates that do not agree with the pronoun gender. Indeed, broad-coverage models of noun gender have proved to be the most important source of world knowledge in automatic pronoun resolution systems.

Previous approaches predict gender by counting the co-occurrence of nouns with pronouns of each gender class. While this provides useful statistics for frequent nouns, many infrequent nouns cannot be classified using this method. Rather than using co-occurrence information directly, we use it to automatically annotate training examples for a large-scale discriminative gender model. Our model collectively classifies all occurrences of a noun in a document using a wide variety of contextual, morphological, and categorical gender features. By leveraging large volumes of unlabeled data, our full semi-supervised system reduces error by 50% over the existing state-of-the-art in gender classification.

1 Introduction

Pronoun resolution is the process of determining which preceding nouns are referred to by a particular pronoun in text. Consider the sentence:

(1) Glen told Glenda that she was wrong about Glendale.

A pronoun resolution system should determine that the pronoun *she* refers to the noun *Glenda*. Pronoun resolution is challenging because it requires a lot of *world knowledge* (general knowledge of word types). If *she* is replaced with the pronoun *he* in (1), *Glen* becomes the antecedent. Pronoun resolution systems need the knowledge of *noun gender* that advises that *Glen* is usually masculine (and thus referred to by *he*) while *Glenda* is feminine.

English third-person pronouns are grouped in four gender/number categories: masculine (*he, his, him, himself*), feminine (*she, her, herself*), neutral (*it, its, itself*), and plural (*they, their, them, themselves*). We broadly refer to these gender and number classes simply as *gender*. The objective of our work is to correctly assign gender to English noun tokens, in context; to determine which class of pronoun will refer to a given noun.

One successful approach to this problem is to build a statistical gender model from a noun's association with pronouns in text. For example, Ge et al. (1998) learn *Ford* has a 94% chance of being neutral, based on its frequent co-occurrence with neutral pronouns in text. Such estimates are noisy but useful. Both Ge et al. (1998) and Bergsma and Lin (2006) show that learned gender is the most important feature in their pronoun resolution systems.

English differs from other languages like French and German in that gender is not an inherent grammatical property of an English noun, but rather a property of a real-world entity that is being referred to. A common noun like *lawyer* can be (semantically) masculine in one document and feminine in another. While previous statistical gender models learn gender for noun types only, we use document context to correctly determine the current gender class of noun tokens, making dynamic decisions on common nouns like *lawyer* and ambiguous names like *Ford*. Furthermore, if a noun type has not yet

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 120–128,
Boulder, Colorado, June 2009.

been observed (an unknown word), previous approaches cannot estimate the gender. Our system, on the other hand, is able to correctly determine that unknown words *corroborators* and *propeller-heads* are plural, while *Pope Formosus* is masculine, using learned contextual and morphological cues.

Our approach is based on the key observation that while gender information from noun-pronoun co-occurrence provides imperfect noun coverage, it can nevertheless provide rich and accurate training data for a large-scale discriminative classifier. The classifier leverages a wide variety of noun properties to *generalize* from the automatically-labeled examples. The steps in our approach are:

1. Training:

(a) Automatically extract a set of seed *(noun,gender)* pairs from high-quality instances in a statistical gender database.
(b) In a large corpus of text, find documents containing these nouns.
(c) For all instances of each noun in each document, create a single, composite feature vector representing all the contexts of the noun in the document, as well as encoding other selected properties of the noun type.
(d) Label each feature vector with the seed noun's corresponding gender.
(e) Train a 4-way gender classifier (masculine, feminine, neutral, plural) from the automatically-labeled vectors.

2. Testing:

(a) Given a new document, create a composite feature vector for all occurrences of each noun.
(b) Use the learned classifier to assign gender to each feature vector, and thus all occurrences of all nouns in the document.

This algorithm achieves significantly better performance than the existing state-of-the-art statistical gender classifier, while requiring no manually-labeled examples to train. Furthermore, by training on a small number of manually-labeled examples, we can combine the predictions of this system with the counts from the original gender database. This semi-supervised extension achieves 95.5% accuracy on final unseen test data, an impressive 50% reduction in error over previous work.

2 Path-based Statistical Noun Gender

Seed *(noun,gender)* examples can be extracted reliably and automatically from raw text, providing the training data for our discriminative classifier. We call these examples *pseudo*-seeds because they are created fully automatically, unlike the small set of manually-created seeds used to initialize other bootstrapping approaches (cf. the bootstrapping approaches discussed in Section 6).

We adopt a statistical approach to acquire the pseudo-seed *(noun,gender)* pairs. All previous statistical approaches rely on a similar observation: if a noun like *Glen* is often referred to by masculine pronouns, like *he* or *his*, then *Glen* is likely a masculine noun. But for most nouns we have no annotated data recording their coreference with pronouns, and thus no data from which we can extract the co-occurrence statistics. Thus previous approaches rely on either hand-crafted coreference-indicating patterns (Bergsma, 2005), or iteratively guess and improve gender models through expectation maximization of pronoun resolution (Cherry and Bergsma, 2005; Charniak and Elsner, 2009). In statistical approaches, the more frequent the noun, the more accurate the assignment of gender.

We use the approach of Bergsma and Lin (2006), both because it achieves state-of-the-art gender classification performance, and because a database of the obtained noun genders is available online.[1] Bergsma and Lin (2006) use an unsupervised algorithm to identify syntactic paths along which a noun and pronoun are highly likely to corefer. To extract gender information, they processed a large corpus of news text, and obtained co-occurrence counts for nouns and pronouns connected with these paths in the corpus. In their database, each noun is listed with its corresponding masculine, feminine, neutral, and plural pronoun co-occurrence counts, e.g.:

glen	555	42	32	34
glenda	8	102	0	11
glendale	24	2	167	18
glendalians	0	0	0	1
glenn	3182	207	95	54
glenna	0	6	0	0

[1] Available at http://www.cs.ualberta.ca/~bergsma/Gender/

This sample of the gender data shows that the noun *glenda*, for example, occurs 8 times with masculine pronouns, 102 times with feminine pronouns, 0 times with neutral pronouns, and 11 times with plural pronouns; 84% of the time *glenda* co-occurs with a feminine pronoun. Note that all nouns in the data have been converted to lower-case.[2]

There are gender counts for 3.1 million English nouns in the online database. These counts form the basis for the state-of-the-art gender classifier. We can either take the most-frequent pronoun-gender (MFPG) as the class (e.g. $feminine$ for *glenda*), or we can supply the logarithm of the counts as features in a 4-way multi-class classifier. We implement the latter approach as a comparison system and refer to it as PATHGENDER in our experiments.

In our approach, rather than using these counts directly, we process the database to automatically extract a high-coverage but also high-quality set of pseudo-seed *(noun,gender)* pairs. First, we filter nouns that occur less than fifty times and whose MFPG accounts for less than 85% of counts. Next, we note that the most reliable nouns should occur relatively often in a coreferent path. For example, note that *importance* occurs twice as often on the web as *Clinton*, but has twenty-four times less counts in the gender database. This is because *importance* is unlikely to be a pronoun's antecedent. We plan to investigate this idea further in future work as a possible filter on antecedent candidates for pronoun resolution. For the present work, simply note that a high ratio of database-count to web-count provides a good indication of the reliability of a noun's gender counts, and thus we filter nouns that have such ratios below a threshold.[3] After this filtering, we have about 45 thousand nouns to which we automatically assign gender according to their MFPG. These *(noun,gender)* pairs provide the seed examples for the training process described in the following section. Figure 1 provides a portion of the ordered $feminine$ seed nouns that we extracted.

[2]Statistical approaches can adapt to the idiosyncrasies of the particular text domain. In the news text from which this data was generated, for example, both the word *ships* and specific instances of ships (*the USS Cole*, *the Titanic*, etc.) are neutral. In Wikipedia, on the other hand, feminine pronouns are often used for ships. Such differences can be learned automatically.

[3]We roughly tuned all the thresholds to obtain the highest number of seeds such that almost all of them looked correct (e.g. Figure 1). Further work is needed to determine whether a different precision/recall tradeoff can improve performance.

```
...
stefanie
steffi graf
steinem
stella mccartney
stellar jayne
stepdaughter
stephanie
stephanie herseth
stephanie white
stepmother
stewardess
...
```

Figure 1: Sample $feminine$ seed nouns

3 Discriminative Learning of Gender

Once we have extracted a number of pseudo-seed *(noun,gender)* pairs, we use them to automatically-label nouns (in context) in raw text. The auto-labeled examples provide training data for discriminative learning of noun gender.

Since the training pairs are acquired from a sparse and imperfect model of gender, what can we gain by training over them? We can regard the Bergsma and Lin (2006) approach and our discriminative system as two orthogonal views of gender, in a co-training sense (Blum and Mitchell, 1998). Some nouns can be accurately labeled by noun-pronoun co-occurrence (a view based on pronoun co-occurrence), and these examples can be used to deduce other gender-indicating regularities (a view based on other features, described below).

We presently explain how examples are extracted using our pseudo-seed pairs, turned into auto-labeled feature vectors, and then used to train a supervised classifier.

3.1 Automatic example extraction

Our example-extraction module processes a large collection of documents (roughly a million documents in our experiments). For each document, we extract all the nouns, including context words within ± 5 tokens of each noun. We then group the nouns by

Class=$masculine$	*String*="Lee"
Contexts =	
"led some to suggest that $\star$, who was born in"	
"$\star$ also downloaded secret files to"	
"$\star$ says he was just making"	
"by mishandling the investigation of $\star$."	
...	

Figure 2: Sample noun training instance

their (lower-case) string. If a group's noun-string is in our set of seed *(noun,gender)* pairs, we assign the corresponding *gender* to be the class of the group. Otherwise, we discard the group. To prevent frequent nouns from dominating our training data, we only keep the first 200 groups corresponding to each noun string. Figure 2 gives an example training noun group with some (selected) context sentences. At test time, all nouns in the test documents are converted to this format for further processing.

We group nouns because there is a strong tendency for nouns to have only one sense (and hence gender) per discourse. We extract contexts because nearby words provide good clues about which gender is being used. The notion that nouns have only one sense per discourse/collocation was also exploited by Yarowsky (1995) in his seminal work on bootstrapping for word sense disambiguation.

3.2 Feature vectors

Once the training instances are extracted, they are converted to labeled feature vectors for supervised learning. The automatically-determined gender provides the class label (e.g., $masculine$ for the group in Figure 2). The features identify properties of the noun and its context that potentially correlate with a particular gender category. We divide the features into two sets: those that depend on the contexts within the document (Context features: features of the *tokens* in the document), and those that depend on the noun string only (Type features). In both cases we induce the feature space from the training examples, keeping only those features that occur more than 5 times.

3.2.1 Context features

The first set of features represent the contexts of the word, using all the contexts in the noun group. To illustrate the potential utility of the context information, consider the context sentences for the masculine noun in Figure 2. Even if these snippets were all the information we were given, it would be easy to guess the gender of the noun.

We use binary attribute-value features to flag, for any of the contexts, the presence of all words at context positions $\pm 1, \pm 2$, etc. (sometimes called *collocation* features (Golding and Roth, 1999)). For example, feature 255920 flags that the word two-to-the-right of the noun is *he*. We also provide features for the presence of all words *anywhere* within ± 5 tokens of the noun (sometimes called *context words*). We also parse the sentence and provide a feature for the noun's parent (and relationship with the parent) in the parse tree. For example, the instance in Figure 2 has features *downloaded(subject)*, *says(subject)*, etc. Since plural nouns should be governed by plural verbs, this feature is likely to be especially helpful for number classification.

3.2.2 Type features

The next group of features represent morphological properties of the noun. Binary features flag the presence of all prefixes and suffixes of one-to-four characters. For multi-token nouns, we have features for the first and last token in the noun. Thus we hope to learn that *Bob* begins masculine nouns while *inc.* ends neutral ones.

Finally, we have features that indicate if the noun or parts of the noun occur on various lists. Indicator features specify if any token occurs on in-house lists of given names, family names, cities, provinces, countries, corporations, languages, etc. A feature also indicates if a token is a corporate designation (like *inc.* or *ltd.*) or a human one (like *Mr.* or *Sheik*). We also made use of the person-name/instance pairs automatically extracted by Fleischman et al. (2003).[4] This data provides counts for pairs such as (Zhang Qiyue, *spokeswoman*) and (Thorvald Stoltenberg, *mediator*). We have features for all *concepts* (like *spokeswoman* and *mediator*) and therefore learn their association with each gender.

3.3 Supervised learning and classification

Once all the feature vectors have been extracted, they are passed to a supervised machine learn-

[4] Available at http://www.mit.edu/~mbf/instances.txt.gz

ing algorithm. We train and classify using a multi-class linear-kernel Support Vector Machine (SVM) (Crammer and Singer, 2001). SVMs are maximum-margin classifiers that achieve good performance on a range of tasks. At test time, nouns in test documents are processed exactly as the training instances described above, converting them to feature vectors. The test vectors are classified by the SVM, providing gender classes for all the nouns in the test document. Since all training examples are labeled automatically (auto-trained), we denote systems using this approach as -AUTO.

3.4 Semi-supervised extension

Although a good gender classifier can be learned from the automatically-labeled examples alone, we can also use a small quantity of gold-standard labeled examples to achieve better performance.

Combining information from our two sets of labeled data is akin to a domain adaptation problem. The gold-standard data can be regarded as high-quality in-domain data, and the automatically-labeled examples can be regarded as the weaker, but larger, out-of-domain evidence.

There is a simple but effective method for combining information from two domains using predictions as features. We train a classifier on the full set of automatically-labeled data (as described in Section 3.3), and then use this classifier's predictions as features in a separate classifier, which is trained on the gold-standard data. This is like the competitive *Feats* domain-adaptation system in Daumé III and Marcu (2006).

For our particular SVM classifier (Section 4.1), predictions take the form of four numerical scores corresponding to the four different genders. Our gold-standard classifier has features for these four predictions plus features for the original path-based gender counts (Section 2).[5] Since this approach uses both automatically-labeled and gold-standard data in a semi-supervised learning framework, we denote systems using this approach as -SEMI.

[5]We actually use 12 features for the path-based counts: the 4 original, and then 4 each for counts for the first and last token in the noun string. See PATHGENDER+ in Section 4.2.

4 Experiments

4.1 Set-up

We parsed the 3 GB AQUAINT corpus (Vorhees, 2002) using Minipar (Lin, 1998) to create our unlabeled data. We process this data as described in Section 3, making feature vectors from the first 4 million noun groups. We train from these examples using a linear-kernel SVM via the the efficient SVM^{multiclass} instance of the SVM^{struct} software package (Tsochantaridis et al., 2004).

To create our gold-standard gender data, we follow Bergsma (2005) in extracting gender information from the anaphora-annotated portion[6] of the American National Corpus (ANC) (Ide and Suderman, 2004). In each document, we first group all nouns with a common lower-case string (exactly as done for our example extraction (Section 3.1)). Next, for each group we determine if a third-person pronoun refers to any noun in that group. If so, we label all nouns in the group with the gender of the referring pronoun. For example, if the pronoun *he* refers to a noun *Brown*, then all instances of *Brown* in the document are labeled as masculine. We extract the genders for 2794 nouns in the ANC training set (in 798 noun groups) and 2596 nouns in the ANC test set (in 642 groups). We apply this method to other annotated corpora (including MUC corpora) to create a development set.

The gold standard ANC training set is used to set the weights on the counts in the PATHGENDER classifiers, and to train the semi-supervised approaches. We also use an SVM to learn these weights. We use the development set to tune the SVM's regularization parameter, both for systems trained on automatically-generated data, and for systems trained on gold-standard data. We also optimize each automatically-trained system on the development set when we include this system's predictions as features in the semi-supervised extension. We evaluate and state performance for all approaches on the final unseen ANC test set.

4.2 Evaluation

The primary purpose of our experiments is to determine if we can improve on the existing state-of-the-art in gender classification (path-based gender

[6]Available at http://www.cs.ualberta.ca/~bergsma/CorefTags/

counts). We test systems both trained purely on automatically-labeled data (Section 3.3), and those that leverage some gold-standard annotations in a semi-supervised setting (Section 3.4). Another purpose of our experiments is to investigate the relative value of our context-based features and type-based features. We accomplish these objectives by implementing and evaluating the following systems:

1. **PATHGENDER**:
   A classifier with the four path-based gender counts as features (Section 2).
2. **PATHGENDER+**:
   A method of back-off to help classify unseen nouns: For multi-token nouns (like *Bob Johnson*), we also include the four gender counts aggregated over all nouns sharing the first token (*Bob* .*), and the four gender counts over all nouns sharing the last token (.* *Johnson*).
3. **CONTEXT-AUTO**:
   Auto-trained system using only context features (Section 3.2.1).
4. **TYPE-AUTO**:
   Auto-trained system using only type features (Section 3.2.2).
5. **FULL-AUTO**:
   Auto-trained system using all features.
6. **CONTEXT-SEMI**:
   Semi-sup. combination of the PATHGENDER+ features and the CONTEXT-AUTO predictions.
7. **TYPE-SEMI**:
   Semi-sup. combination of the PATHGENDER+ features and the TYPE-AUTO predictions.
8. **FULL-SEMI**:
   Semi-sup. combination of the PATHGENDER+ features and the FULL-AUTO predictions.

We evaluate using *accuracy*: the percentage of labeled nouns that are correctly assigned a gender class. As a baseline, note that always choosing *neutral* achieves 38.1% accuracy on our test data.

5 Results and Discussion

5.1 Main results

Table 1 provides our experimental results. The original gender counts already do an excellent job classifying the nouns; PATHGENDER achieves 91.0% accuracy by looking for exact noun matches. Our simple method of using back-off counts for the first and last token, PATHGENDER+, achieves 92.1%. While PATHGENDER+ uses gold standard data to determine optimum weights on the twelve counts, FULL-AUTO achieves 92.6% accuracy using no gold standard training data. This confirms that our algorithm, using no manually-labeled training data, can produce a competitive gender classifier.

1. PATHGENDER	91.0
2. PATHGENDER+	92.1
3. CONTEXT-AUTO	79.1
4. TYPE-AUTO	89.1
5. FULL-AUTO	92.6
6. CONTEXT-SEMI	92.4
7. TYPE-SEMI	91.3
8. FULL-SEMI	**95.5**

Table 1: Noun gender classification accuracy (%)

Both PATHGENDER and PATHGENDER+ do poorly on the noun types that have low counts in the gender database, achieving only 63% and 66% on nouns with less than ten counts. On these same nouns, FULL-AUTO achieves 88% performance, demonstrating the robustness of the learned classifier on the most difficult examples for previous approaches (FULL-SEMI achieves 94% on these nouns).

If we break down the contribution of the two feature types in FULL-AUTO, we find that we achieve 89.1% accuracy by only using type features, while we achieve 79.1% with only context features. While not as high as the type-based accuracy, it is impressive that almost four out of five nouns can be classified correctly based purely on the document context, using no information about the noun itself. This is information that has not previously been systematically exploited in gender classification models.

We examine the relationship between training data size and accuracy by plotting a (logarithmic-scale) learning curve for FULL-AUTO (Figure 3). Although using four million noun groups originally seemed sufficient, performance appears to still be increasing. Since more training data can be generated automatically, it appears we have not yet reached the full power of the FULL-AUTO system. Of course, even with orders of magnitude more data, the system

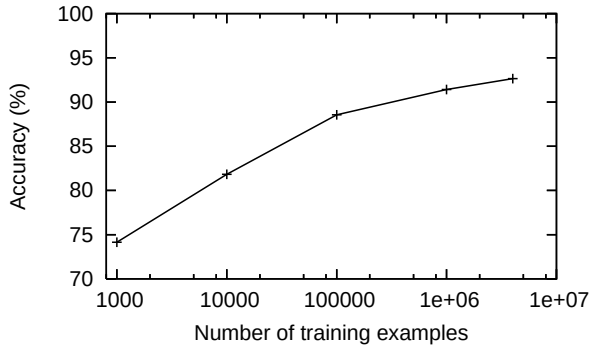


Figure 3: Noun gender classification learning curve for FULL-AUTO

does not appear destined to reach the performance obtained through other means described below.

We achieve even higher accuracy when the output of the -AUTO systems are combined with the original gender counts (the semi-supervised extension). The relative value of the context and type-based features is now reversed: using only context-based features (CONTEXT-SEMI) achieves 92.4%, while using only type-based features (TYPE-SEMI) achieves 91.3%. This is because much of the type information is already implicit in the PATHGENDER counts. The TYPE-AUTO predictions contribute little information, only fragmenting the data and leading to over-training and lower accuracy. On the other hand, the CONTEXT-AUTO predictions improve accuracy, as these scores provide orthogonal and hence helpful information for the semi-supervised classifier.

Combining FULL-AUTO with our enhanced path gender counts, PATHGENDER+, results in the overall best performance, 95.5% for FULL-SEMI, significantly better than PATHGENDER+ alone.[7] This is a 50% error reduction over the PATHGENDER system, strongly confirming the benefit of our semi-supervised approach.

To illustrate the importance of the unlabeled data, we created a system that uses all features, including the PATHGENDER+ counts, and trained this system using only the gold standard training data. This system was unable to leverage the extra features to improve performance; its accuracy was 92.0%, roughly equal to PATHGENDER+ alone. While SVMs work well with high-dimensional data, they simply cannot exploit features that do not occur in the training set.

5.2 Further improvements

We can improve performance further by doing some simple coreference before assigning gender. Currently, we only group nouns with the same string, and then decide gender collectively for the group. There are a few cases, however, where an ambiguous surname, such as *Willey*, can only be classified correctly if we link the surname to an earlier instance of the full name, e.g. *Katherine Willey*. We thus added the following simple post-processing rule: If a noun is classified as $masculine$ or $feminine$ (like the ambiguous *Willey*), and it was observed earlier as the last part of a larger noun, then re-assign the gender to $masculine$ or $feminine$ if one of these is the most common path-gender count for the larger noun. We back off to counts for the first name (e.g. *Kathleen* .*) if the full name is unobserved.

This enhancement improved the PATHGENDER and PATHGENDER+ systems to 93.3% and 94.3%, respectively, while raising the accuracy of our FULL-SEMI system to 96.7%. This demonstrates that the surname-matching post-processor is a simple but worthwhile extension to a gender predictor.[8]

The remaining errors represent a number of challenging cases: *United States*, *group*, and *public* labeled as $plural$ but classified as $neutral$; *spectator* classified as $neutral$, etc. Some of these may yield to more sophisticated joint classification of coreference and gender, perhaps along the lines of work in named-entity classification (Bunescu and Mooney, 2004) or anaphoricity (Denis and Baldridge, 2007).

While gender has been shown to be the key feature for statistical pronoun resolution (Ge et al., 1998; Bergsma and Lin, 2006), it remains to be seen whether the exceptional accuracy obtained here will translate into improvements in resolution performance. However, given the clear utility of gender in coreference, substantial error reductions in gender

[7] We evaluate significance using McNemar's test, $p<0.01$. Since McNemar's test assumes independent classifications, we apply the test to the classification of noun *groups*, not instances.

[8] One might wonder, why not provide special features so that the system can *learn* how to handle ambiguous nouns that occurred as sub-phrases in earlier names? The nature of our training data precludes this approach. We only include *un*ambiguous examples as pseudo-seeds in the learning process. Without providing ambiguous (but labeled) surnames in some way, the learner will not take advantage of features to help classify them.

assignment will likely be a helpful contribution.

6 Related Work

Most coreference and pronoun resolution papers mention that they use gender information, but few explain how it is acquired. Kennedy and Boguraev (1996) use gender information produced by their enhanced part-of-speech tagger. Gender mistakes account for 35% of their system's errors. Gender is less crucial in some genres, like computer manuals; most nouns are either neutral or plural and gender can be determined accurately based solely on morphological information (Lappin and Leass, 1994).

A number of researchers (Evans and Orăsan, 2000; Soon et al., 2001; Harabagiu et al., 2001) use WordNet classes to infer gender knowledge. Unfortunately, manually-constructed databases like WordNet suffer from both low coverage and rare senses. Pantel and Ravichandran (2004) note that the nouns *computer* and *company* both have a WordNet sense that is a hyponym of *person*, falsely indicating these nouns would be compatible with pronouns like *he* or *she*. In addition to using WordNet classes, Soon et al. (2001) assign gender if the noun has a gendered designator (like *Mr.* or *Mrs.*) or if the first token is present on a list of common human first names. Note that we incorporate such contextual and categorical information (among many other information sources) automatically in our discriminative classifier, while they manually specify a few high-precision rules for particular gender cues.

Ge et al. (1998) pioneered the statistical approach to gender determination. Like others, they consider gender and number separately, only learning statistical gender for the masculine, feminine, and neutral classes. While gender and number can be handled together for pronoun resolution, it might be useful to learn them separately for other applications. Other statistical approaches to English noun gender are discussed in Section 2.

In languages with 'grammatical' gender and plentiful gold standard data, gender can be tagged along with other word properties using standard supervised tagging techniques (Hajič and Hladká, 1997). While our approach is the first to exploit a dual or orthogonal representation of English noun gender, a bootstrapping approach has been applied to determining grammatical gender in other languages by Cucerzan and Yarowsky (2003). In their work, the two orthogonal views are: 1) the context of the noun, and 2) the noun's morphological properties. Bootstrapping with these views is possible in other languages where context is highly predictive of gender class, since contextual words like adjectives and determiners inflect to agree with the grammatical noun gender. We initially attempted a similar system for English noun gender but found context alone to be insufficiently predictive.

Bootstrapping is also used in general information extraction. Brin (1998) shows how to alternate between extracting instances of a class and inducing new instance-extracting patterns. Collins and Singer (1999) and Cucerzan and Yarowsky (1999) apply bootstrapping to the related task of named-entity recognition. Our approach was directly influenced by the hypernym-extractor of Snow et al. (2005) and we provided an analogous summary in Section 1. While their approach uses WordNet to label hypernyms in raw text, our initial labels are generated automatically. Etzioni et al. (2005) also require no labeled data or hand-labeled seeds for their named-entity extractor, but by comparison their classifier only uses a very small number of both features and automatically-generated training examples.

7 Conclusion

We have shown how noun-pronoun co-occurrence counts can be used to automatically annotate the gender of millions of nouns in unlabeled text. Training from these examples produced a classifier that clearly exceeds the state-of-the-art in gender classification. We incorporated thousands of useful but previously unexplored indicators of noun gender as features in our classifier. By combining the predictions of this classifier with the original gender counts, we were able to produce a gender predictor that achieves 95.5% classification accuracy on 2596 test nouns, a 50% reduction in error over the current state-of-the-art. A further name-matching post-processor reduced error even further, resulting in 96.7% accuracy on the test data. Our final system is the broadest and most accurate gender model yet created, and should be of value to many pronoun and coreference resolution systems.

References

Shane Bergsma and Dekang Lin. 2006. Bootstrapping path-based pronoun resolution. In *COLING-ACL*, pages 33–40.

Shane Bergsma. 2005. Automatic acquisition of gender information for anaphora resolution. In *Canadian Conference on Artificial Intelligence*, pages 342–353.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*, pages 92–100.

Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *WebDB Workshop at 6th International Conference on Extending Database Technology*, pages 172–183.

Razvan Bunescu and Raymond J. Mooney. 2004. Collective information extraction with relational Markov networks. In *ACL*, pages 438–445.

Eugene Charniak and Micha Elsner. 2009. EM works for pronoun anaphora resolution. In *EACL*.

Colin Cherry and Shane Bergsma. 2005. An expectation maximization approach to pronoun resolution. In *CoNLL*, pages 88–95.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *EMNLP-VLC*, pages 100–110.

Koby Crammer and Yoram Singer. 2001. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292.

Silviu Cucerzan and David Yarowsky. 1999. Language independent named entity recognition combining morphological and contextual evidence. In *EMNLP-VLC*, pages 90–99.

Silviu Cucerzan and David Yarowsky. 2003. Minimally supervised induction of grammatical gender. In *NAACL*, pages 40–47.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Pascal Denis and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference using integer programming. In *NAACL-HLT*, pages 236–243.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165(1):91–134.

Richard Evans and Constantin Orăsan. 2000. Improving anaphora resolution by identifying animate entities in texts. In *DAARC*, pages 154–162.

Michael Fleischman, Eduard Hovy, and Abdessamad Echihabi. 2003. Offline strategies for online question answering: answering questions before they are asked. In *ACL*, pages 1–7.

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.

Andrew R. Golding and Dan Roth. 1999. A Winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

Jan Hajič and Barbora Hladká. 1997. Probabilistic and rule-based tagger of an inflective language: a comparison. In *ANLP*, pages 111–118.

Sanda Harabagiu, Razvan Bunescu, and Steven Maiorano. 2001. Text and knowledge mining for coreference resolution. In *NAACL*, pages 55–62.

Nancy Ide and Keith Suderman. 2004. The American National Corpus first release. In *LREC*, pages 1681–84.

Christopher Kennedy and Branimir Boguraev. 1996. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *COLING*, pages 113–118.

Shalom Lappin and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *LREC Workshop on the Evaluation of Parsing Systems*.

Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *HLT-NAACL*, pages 321–328.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*, pages 1297–1304.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *ICML*.

Ellen Vorhees. 2002. Overview of the TREC 2002 question answering track. In *Proceedings of the Eleventh Text REtrieval Conference (TREC)*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, pages 189–196.

Improving Translation Lexicon Induction from Monolingual Corpora via Dependency Contexts and Part-of-Speech Equivalences

Nikesh Garera, Chris Callison-Burch, David Yarowsky
Department of Computer Science, Johns Hopkins University
Baltimore MD, USA
{ngarera,ccb,yarowsky}@cs.jhu.edu

Abstract

This paper presents novel improvements to the induction of translation lexicons from monolingual corpora using multilingual dependency parses. We introduce a dependency-based context model that incorporates long-range dependencies, variable context sizes, and reordering. It provides a 16% relative improvement over the baseline approach that uses a fixed context window of adjacent words. Its Top 10 accuracy for noun translation is higher than that of a statistical translation model trained on a Spanish-English parallel corpus containing 100,000 sentence pairs. We generalize the evaluation to other word-types, and show that the performance can be increased to 18% relative by preserving part-of-speech equivalencies during translation.

1 Introduction

Recent trends in machine translation illustrate that highly accurate word and phrase translations can be learned automatically given enough parallel training data (Koehn et al., 2003; Chiang, 2007). However, large parallel corpora exist for only a small fraction of the world's languages, leading to a bottleneck for building translation systems in low-density languages such as Swahili, Uzbek or Punjabi. While parallel training data is uncommon for such languages, more readily available resources include small translation dictionaries, comparable corpora, and large amounts of monolingual data.

The marked difference in the availability of monolingual vs parallel corpora has led several researchers to develop methods for automatically learning bilingual lexicons, either by using monolingual corpora (Rapp, 1999; Koehn and Knight, 2002; Schafer and Yarowsky, 2002; Haghighi et al., 2008) or by exploiting the cross-language evidence of closely related "bridge" languages that have more resources (Mann and Yarowsky, 2001).

This paper investigates new ways of learning translations from monolingual corpora. We extend the Rapp (1999) model of context vector projection using a seed lexicon. It is based on the intuition that translations will have similar lexical context, even in unrelated corpora. For example, in order to translate the word "airplane", the algorithm builds a context vector which might contain terms such as "passengers", "runway", "airport", etc. and words in target language that have their translations (obtained via seed lexicon) in surrounding context can be considered as likely translations. We extend the basic approach by formulating a context model that uses dependency trees. The use of dependencies has the following advantages:

- Long distance dependencies allow associated words to be included in the context vector even if they fall outside of the fixed-window used in the baseline model.
- Using relationships like parent and child instead of absolute positions alleviates problems when projecting vectors between languages with different word orders.
- It achieves better performance than baseline context models across the board, and better performance than statistical translation models on Top-10 accuracy for noun translation when trained on identical data.

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 129–137,
Boulder, Colorado, June 2009.

We further show that an extension based on part-of-speech clustering can give similar accuracy gains for learning translations of all word-types, deepening the findings of previous literature which mainly focused on translating nouns (Rapp, 1999; Koehn and Knight, 2002; Haghighi et al., 2008).

2 Related Work

The literature on translation lexicon induction for low-density languages falls in to two broad categories: 1) Effectively utilizing similarity between languages by choosing a high-resource "bridge" language for translation (Mann and Yarowsky, 2001; Schafer and Yarowsky, 2002) and 2) Extracting noisy clues (such as similar context) from monolingual corpora with help of a seed lexicon (Rapp, 1999; Koehn and Knight, 2002; Schafer and Yarowsky, 2002, Haghighi et al., 2008). The latter category is more relevant to this work and is explained in detail below.

The idea of words with similar meaning having similar contexts in the same language comes from the Distributional Hypothesis (Harris, 1985) and Rapp (1999) was the first to propose using context of a given word as a clue to its translation. Given a German word with an unknown translation, a German context vector is constructed by counting its surrounding words in a monolingual German corpus. Using an incomplete bilingual dictionary, the counts of the German context words with known translations are projected onto an English vector. The projected vector for the German word is compared to the vectors constructed for all English words using a monolingual English corpus. The English words with the highest vector similarity are treated as translation candidates. The original work employed a relatively large bilingual dictionary containing approximately 16,000 words and tested only on a small collection of 100 manually selected nouns.

Koehn and Knight (2002) tested this idea on a larger test set consisting of the 1000 most frequent words from a German-English lexicon. They also incorporated clues such as frequency and orthographic similarity in addition to context. Schafer and Yarowsky, (2002) independently proposed using frequency, orthographic similarity and also showed improvements using temporal and word-burstiness similarity measures, in addition to context. Haghighi et al., (2008) made use of contextual and orthographic clues for learning a generative model from monolingual corpora and a seed lexicon.

All of the aforementioned work defines context similarity in terms of the adjacent words over a window of some arbitary size (usually 2 to 4 words), as initially proposed by Rapp (1999). We show that the model for surrounding context can be improved by using dependency information rather than strictly relying on adjacent words, based on the success of dependency trees for monolingual clustering and disambiguation tasks (Lin and Pantel, 2002; Pado and Lapata, 2007) and the recent developments in multilingual dependency parsing literature (Buchholz and Marsi, 2006; Nivre et al., 2007).

We further differentiate ourselves from previous work by conducting a second evaluation which examines the accuracy of translating all word types, rather than just nouns. While the straightforward application of context-based model gives a lower overall accuracy than nouns alone, we show how learning a mapping of part-of-speech tagsets between the source and target language can result in comparable performance to that of noun translation.

3 Translation by Context Vector Projection

This section details how translations are discovered from monolingual corpora through context vector projection. Section 3.1 defines alternative ways of modeling context vectors, and including baseline models and our dependency-based model.

The central idea of Rapp's method for learning translations is that of context vector projection and vector similarity. The goodness of semantic "fit" of candidate translations is measured as the vector similarity between two words. Those vectors are drawn from two different languages, so the vector for one word must first be projected onto the language space of the other. The algorithm for creating, projecting and comparing vectors is described below, and illustrated in Figure 1.

Algorithm:

1. Extract context vectors:

   Given a word in source language, say s_w, create a vector using the surrounding context words and call this reference source vector rs_{s_w} for

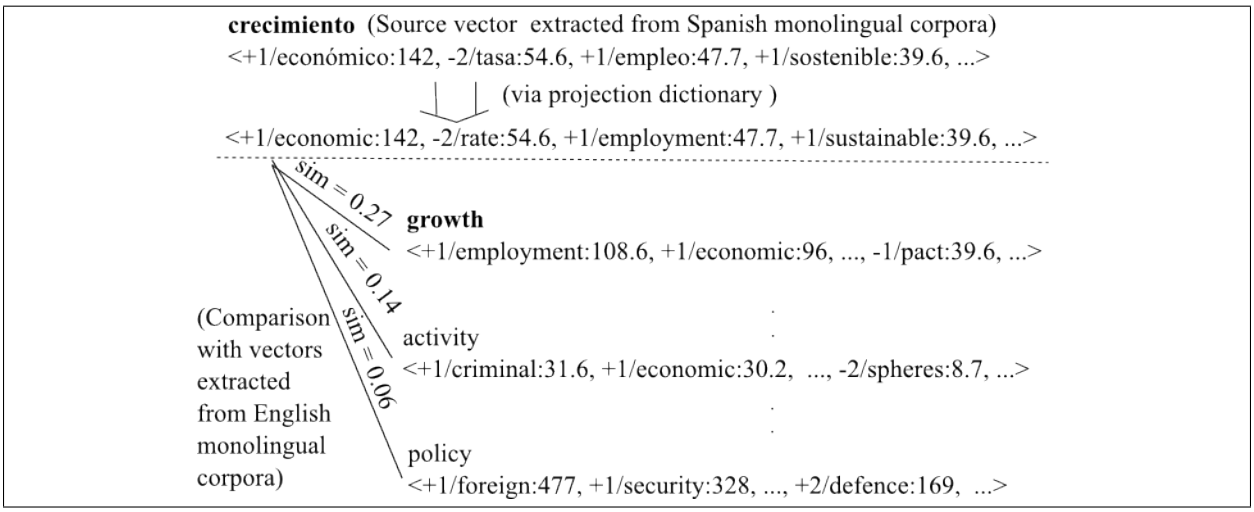


Figure 1: Illustration of (Rapp, 1999) model for translating spanish word "crecimiento (growth)" via dependency context vectors extracted from respective monolingual corpora as explained in Section 3.1.2

source word s_w. The actual composition of this vector varies depending on how the surrounding context is modeled. The context model is independent of the algorithm, and various models are explained in later sections.

2. Project reference source vector:

   Project all the source vector words contained in the projection dictionary onto the vector space for the target language, retaining the counts from source corpus. This vector now exists in the target language space and is called the reference target vector rt_{s_w}. This vector may be sparse, depending on how complete the bilingual dictionary is, because words without dictionary entires will receive zero counts in the reference target vector.

3. Rank candidates by vector similarity:

   For each word t_{w_i} in the target language a context vector is created using the target language monolingual corpora as in Step 1. Compute a similarity score between the context vector of $t_{w_i} = \langle c_{i1}, c_{i2}, ..., c_{in} \rangle$ and reference target vector $rt_{s_w} = \langle r_1, r_2, ..., r_n \rangle$. The word with the maximum similarity score $t^*_{w_i}$ is chosen as the candidate translation of s_w.

   The vector similarity can be computed in a number of ways. Our setup we used cosine similarity:

$$t^*_{w_i} = \text{argmax}_{t_{w_i}} \frac{c_{i1} \cdot r_1 + c_{i2} \cdot r_2 + ... + c_{in} \cdot r_n}{\sqrt{c_{i1}^2 + c_{i2}^2 + ... + c_{in}^2} \sqrt{r_1^2 + r_2^2 + ... + r_n^2}}$$

   Rapp (1999) used l1-norm metric after normalizing the vectors to unit length, Koehn and Knight (2002) used Spearman rank order correlation, and Schafer and Yarowsky (2002) use cosine similarity. We found that cosine similarity gave the best results in our experimental conditions. Other similarity measures may be used equally well.

3.1 Models of Context

We compared several context models. Empirical results for their ability to find accurate translations are given in Section 5.

3.1.1 Baseline model

In the baseline model, the context is computed using adjacent words as in (Rapp,1999; Koehn and Knight, 2002; Schafer and Yarowsky, 2002; Haghighi et al., 2008). Given a word in source language, say s_w, count all its immediate context words appearing in a window of four words. The counts are collected seperately for each position by keeping track of four seperate vectors for positions -2, -1, +1 and +2. Thus each vector is a sparse vector, having the # of dimensions as the size of source language vocabulary. Each dimension is also reweighted by multiplying the inverse document frequency (IDF)

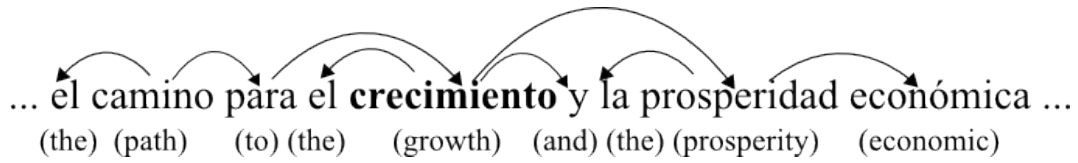


Figure 2: Illustration of using dependency trees to model richer contexts for projection

as in the standard TF.IDF weighting scheme[1]. These vectors are then concatenated into a single vector, having dimension four times the size of the vocabulary. This vector is called the reference source vector rs_{s_w} for source word s_w.

3.1.2 Modeling context using dependency trees

We use dependency parsing to extend the context model. Our context vectors use contexts derived from head-words linked by dependency trees instead of using the immediate adjacent lexical words. The use of dependency trees for modeling contexts has been shown to help in monolingual clustering tasks of finding words with similar meaning (Lin and Pantel, 2002) and we show how they can be effectively used for translation lexicon induction.

Position	Adjacent Context	Dependency Context
-2	para	camino
-1	el	para
+1	y	prosperidad, y, el
+2	la	económica

Table 1: Contrasting context words derived from the adjacent vs dependency models for the above example

The four vectors for positions -1, +1, -2 and +2 in the baseline model get mapped to immediate parent (-1), immediate child (+1), grandparent (-2) and grandchild (+2). An example of using the dependency tree context is shown in Figure 2, and the dependency context is shown in contrast with the adjacent context in Table 1, showing the selection of more salient words by using the dependencies.

Note that while we are limiting to four positions in the tree, it does not imply that only a maximum of four context words are selected since the word can have multiple immediate children depending upon the dependency parse of the sentence. Hence, this approach allows for a dynamic context size, with the number of context words varying with the number of children and parents at the two levels.

Another advantage of this method is that it alleviates the reordering problem as we use tree positions (consisting of head-words) as compared to the adjacent position in the baseline context model. For example, if the source spanish word to be translated was "prosperidad", then in the example shown in Figure 2, in case of adjacent context, the context word "económica" will show up in +1 position in Spanish and -1 position in English (as adjectives come before nouns in English) but in case of dependency context, the adjective will be the child of noun and hence will show up in +1 position in both languages. Thus, we do not need to use a bag of word model as in Section 3 in order to avoid learning the explicit mapping that adjectives and nouns in Spanish and English are reversed.

4 Experimental Design

For our initial set of experiments we compared several different vector-based context models:

- Adj_{bow} – A baseline model which used bag of words model with a fixed window of 4 words, two on either side of the word to be translated.
- Adj_{posn} – A second baseline that used a fixed window of 4 words but which took positional into account.
- Dep_{bow} – A dependency model which did not distinguish between grandparent, parent, child and grandparent relations, analogous to the bag of words model.
- Dep_{posn} – A dependency model which did include such relationships, and was analogous to the position-based baseline.
- $\text{Dep}_{\text{posn + rev}}$ – The above Dep_{posn} model applied in both directions (Spanish-to-English and English-to-Spanish) using their sum as the final translation score.

We contrasted the accuracy of the above methods, which use monolingual corpora, with a statistical

[1] In order to compute the IDF, while there were no clear document boundaries in our corpus, a virtual document boundary was created by binning after every 1000 words.

model trained on bilingual parallel corpora. We refer to that model as $\text{Moses}_{\text{en-es-100k}}$, because it was trained using the Moses toolkit (Koehn et al., 2007).

4.1 Training Data

All context models were trained on a Spanish corpus containing 100,000 sentences with 2.13 million words and an English corpus containing 100,000 sentences with 2.07 million words. The Spanish corpus was parsed using the MST dependency parser (McDonald et al., 2005) trained using dependency trees generated from the the English Penn Treebank (Marcus et al., 1993) and Spanish CoNLL-X data (Buchholz and Marsi, 2006).

So that we could directly compare against statistical translation models, our Spanish and English monolingual corpora were drawn from the Europarl parallel corpus (Koehn, 2005). The fact that our two monolingual corpora are taken from a parallel corpus ensures that the assumption that similar contexts are a good indicator of translation holds. This assumption underlies in all work of translation lexicon induction from comparable monolingual corpora, and here we strongly bias toward that assumption. Despite the bias, the comparison of different context models holds, since all models are trained on the same data.

4.2 Evaluation Criterion

The models were evaluated in terms of exact-match translation accuracy of the 1000 most frequent nouns in a English-Spanish dictionary. The accuracy was calculated by counting how many mappings exactly match one of the entries in the dictionary. This evaluation criterion is similar to the setup used by Koehn and Knight (2002). We compute the Top N accuracy in the standard way as the number of Spanish test words whose Top N English translation candidates contain a lexicon translation entry out of the total number of Spanish words that can be mapped correctly using the lexicon entries. Thus if "crecimiento, growth" is the correct mapping based on the lexicon entries, the translation for "crecimiento" will be counted as correct if "growth" occurs in the Top N English translation candidates for "crecimiento".

Note that the exact-match accuracy is a conservative estimate as it is possible that the algorithm may propose a reasonable translation for the given

camino			
Dep_{posn} Cntxt	Model	Adj_{bow} Cntxt	Model
way	0.124	intentions	0.22
solution	0.097	**way**	0.21
steps	0.094	idea	0.20
path	0.093	thing	0.20
debate	0.085	faith	0.18
account	0.082	**steps**	0.17
means	0.080	example	0.17
work	0.079	news	0.16
approach	0.074	work	0.16
issue	0.073	attitude	0.15

Table 2: Top 10 translation candidates for the spanish word "camino (way)" for the best adjacent context model (Adj_{bow}) and best dependency context model (Dep_{posn}). The bold English terms show the acceptable translations.

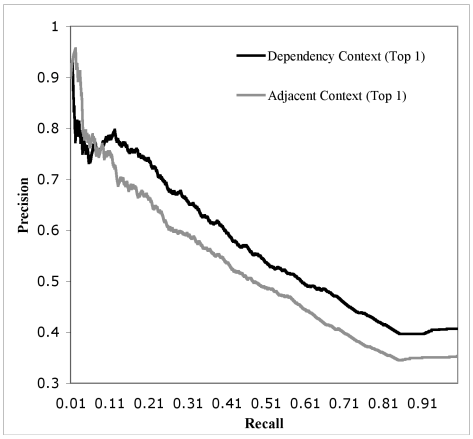


Figure 3: Precision/Recall curve showing superior performance of dependency context model as compared to adjacent context at different recall points. Precision is the fraction of tested Spanish words with Top 1 translation correct and Recall is fraction of the 1000 Spanish words tested upon.

Spanish word but is marked incorrect if it does not exist in the lexicon. Because it would be intractable to compare each projected vector against the vectors for all possible English words, we limited ourselves to comparing the projected vector from each Spanish word against the vectors for the 1000 most frequent English nouns, following along the lines of previous work (Koehn and Knight, 2002; Haghighi et al., 2008).

5 Results

Table 3 gives the Top 1 and Top 10 accuracy for each of the models on their ability to translate Spanish nouns into English. Examples of the top 10 translations using the best performing baseline and dependency-based models are shown in Table 2. The baseline models Adj_{posn} and Adj_{bow} differ in that the

Model	$\text{Acc}_{\text{Top 1}}$	$\text{Acc}_{\text{Top 10}}$
Adj_{bow}	35.3%	59.8%
Adj_{posn}	20.9%	46.9%
Dep_{bow}	41.0%	62.0%
Dep_{posn}	41.0%	64.1%
$\text{Dep}_{\text{posn + rev}}$	42.9%	**65.5%**
$\text{Moses}_{\text{en-es-100k}}$	**56.4%**	62.7%

Table 3: Performance of various context-based models learned from monolingual corpora and phrase-table learned from parallel corpora on Noun translation.

latter disregards the position information in the context vector and simply uses a bag of words instead. Table 3 shows that Adj_{bow} gains using this simplification. A bag of words vector approach pools counts together, which helps to reduce data sparsity. In the position based model the vector is four times as long. Additionally, the bag of words model can help when there is local re-ordering between the two languages. For instance, Spanish adjectives often follow nouns whereas in English the the ordering is reversed. Thus, one can either learn position mappings, that is, position +1 for adjectives in Spanish is the same as position -1 in English or just add the the word counts from different positions into one common vector as considered in the bag of words approach.

Using dependency trees also alleviates the problem of position mapping between source and target language. Table 3 shows the performance using the dependency based models outperforms the baseline models substantially. Comparing Dep_{bow} to Dep_{posn} shows that ignoring the tree depth and treating it as a bag of words does not increase the performance. This contrasts with the baseline models. The dependency positions account for re-ordering automatically. The precision-recall curve in Figure 3 shows that the dependency-based context performs better than adjancet context at almost all recall levels.

The $\text{Moses}_{\text{en-es-100k}}$ model shows the performance of the statistical translation model trained on a bilingual parallel corpus. While the system performs best in Top 1 accuracy, the dependency context-based model that ignores the sentence alignments surprisingly performs better in case of Top 10 accuracy, showing substantial promise.

While computing the accuracy using the phrase-table learned from parallel corpora ($\text{Moses}_{\text{en-es-100k}}$), the translation probabilities from both directions ($p(es|en)$ and $p(en|es)$) were used to rank the candidates. We also apply the monolingual context-based model in the reverse direction (from English to Spanish) and the row with label $\text{Dep}_{\text{posn + rev}}$ in Table 3 shows further gains using both directions.

Spanish	English	Sim Score	Is present in lexicon
señores	gentlemen	0.99	NO
xenofobia	xenophobia	0.87	YES
diversidad	diversity	0.73	YES
chipre	cyprus	0.66	YES
mujeres	women	0.65	YES
alemania	germany	0.65	YES
explotación	exploitation	0.63	YES
hombres	men	0.62	YES
república	republic	0.60	YES
racismo	racism	0.59	YES
comercio	commerce	0.58	YES
continente	continent	0.53	YES
gobierno	government	0.52	YES
israel	israel	0.52	YES
francia	france	0.52	YES
fundamento	certainty	0.51	NO
suecia	sweden	0.50	YES
tráfico	space	0.49	NO
televisión	tv	0.48	YES
francesa	portuguese	0.48	NO

Table 4: List of 20 most confident mappings using the dependency context based model for noun translation. Note that although the first mapping is the correct one, it was not present in the lexicon used for evaluation and hence is marked as incorrect.

6 Further Extensions: Generalizing to other word types via tagset mapping

Most of the previous literature on this problem focuses on evaluating on nouns (Rapp, 1999; Koehn and Knight 2002; Haghighi et al., 2008). However the vector projection approach is general, and should be applicable to other word-types as well. We evaluated the models with new test set containing 1000 most frequent words (not just nouns) in the English-Spanish lexicon.

We used the dependency-based context model to create translations for this new set. The row labeled Dep_{posn} in Table 5 shows that the accuracy on this set is lower when compared to evaluating only on nouns. The main reason for lower accuracy is that closed class words are often the most frequent and tend to have a wide range of contexts resulting in reasonable translation for most words include open class words via the context model. For instance, the English preposition "to" appears as the most confident translation for 147 out of the 1000 Spanish test

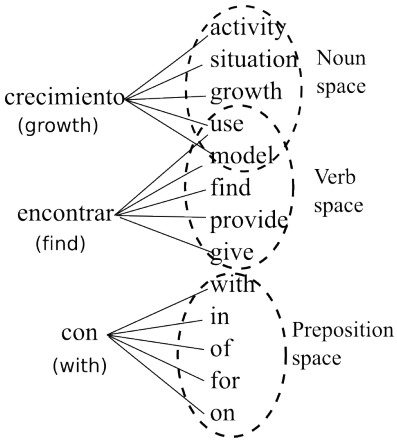


Figure 4: Illustration of using part-of-speech tag mapping to restrict candidate space of translations

words and in none (rightly so) after restricting the translations by part-of-speech categories.

This problem can be greatly reduced by making use of the intuition that part-of-speech is often preserved in translation, thus the space of possible candidate translation can be largely reduced based on the part-of-speech restrictions. For example, a noun in source language will usually be translated as noun in target language, determiner will be translated as determiner and so on. This idea is more clearly illustrated in in Figure 4. We do not impose a hard restriction but rather compute a ranking based on the conditional probability of candidate translation's part-of-speech tag given source word's tag.

An interesting problem in using part-of-speech restrictions is that corpora in different languages have been tagged using widely different tagsets and the following subsection explains this problem in detail:

6.1 Mapping Part-of-Speech tagsets in different languages

The English tagset was derived from the Penn treebank consisting of 53 tags (including punctuation markers) and the Spanish tagset was derived from the Cast3LB dataset consisting of 57 tags but there is a large difference in the morphological and syntactic features marked by the tagset. For example, the Spanish tagset as different tags for masculine and feminine nouns and also has a different tag for coordinated nouns, all of which need to be mapped to the singular or plural noun category available in English tagset. Figure 5 shows an illustration of the mapping problem between the Spanish and English POS tags.

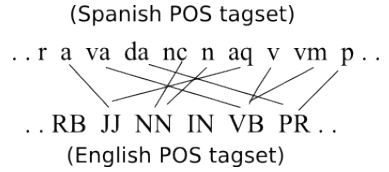


Figure 5: Illustration of mapping Spanish part-of-speech tagset to English tagset. The tagsets vary greatly in notation and the morphological/syntactic constituents represented and need to be mapped first, using the algorithm described in Section 6.1.

We now describe an empirical approach for learning the mapping between tagsets using the English-Spanish projection dictionary used in the monolingual context-based models for translation. Given a small English-Spanish bilingual dictionary and a n-best list of part-of-speech tags for each word in the dictionary[2], we compute conditional probability of translating a source word with pos tag s_{pos_i} to a target with pos tag t_{pos_j} as follows:

$$p(t_{pos_j}|s_{pos_i}) = \frac{c(s_{pos_i}, t_{pos_j})}{c(s_{pos_i})} = \frac{\sum_{s_w \in S, t_w \in T} p(s_{pos_i}|s_w) \cdot p(t_{pos_j}|t_w) \cdot I_{dict}(s_w, t_w)}{\sum_{s_w \in S} p(s_{pos_i}|s_w)}$$

where

- S and T are the source and target vocabulary in the seed dictionary, with s_w and t_w being any of the words in the respective sets.
- $p(s_{pos_i}|s_w), p(t_{pos_j}|t_w)$ are obtained using relative frequencies in a part-of-speech tagged corpus in the source/target languages respectively, and are used as soft counts.
- $I_{dict}(s_w, t_w)$ is the indicator function with value 1 if the pair (s_w, t_w) occurs in the seed dictionary and 0 otherwise.

In essence, the mapping between tagsets is learned using the known translations from a small dictionary.

Given a source word s_w to translate, its most likely tag s^*_{pos}, and the most likely mapping of this tag into English t^*_{pos} computed as above, the translation candidates with part-of-speech tag t^*_{pos} are considered for comparison with vector similarity and

[2]The n-best part-of-speech tag list for any word in the dictionary was derived using the relative frequencies in a part-of-speech annotated corpora in the respective languages

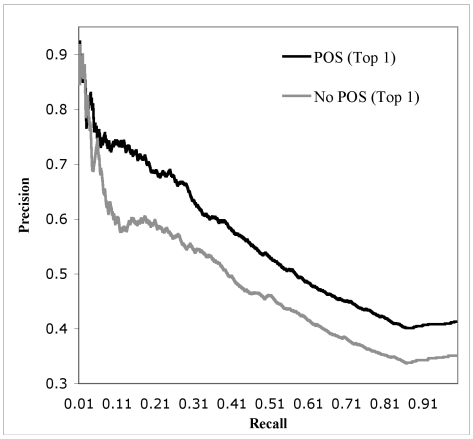


Figure 6: Precision/Recall curve showing superior performance of using part-of-speech equivalences for translating all word-types. Precision is the fraction of tested Spanish words with Top 1 translation correct and Recall is fraction of the 1000 Spanish words tested upon.

the other candidates with $t_{pos_j} \neq t^*_{pos}$ are discarded from the candidate space. Figure 4 shows an example of restricting the candidate space using POS tags.

Model	$\text{Acc}_{\text{Top 1}}$	$\text{Acc}_{\text{Top 10}}$
Dep_{posn}	35.1%	62.9%
+ POS	**41.3%**	**66.4%**

Table 5: Performance of dependency context-based model along with addition of part-of-speech mapping model on translating all word-types.

The row labeled $+POS$ in Table 5 shows the part-of-speech tags provides substantial gain as compared to direct application of dependency context-based model and is also comparable to the accuracy obtained evaluating just on nouns in Table 3.

7 Conclusion

This paper presents a novel contribution to the standard context models used when learning translation lexicons from monolingual corpora by vector projection. We show that using contexts based on dependency parses can provide more salient contexts, allow for dynamic context size, and account for word reordering in the source and target language. An exact-match evaluation shows 16% relative improvement by using a dependency-based context model over the standard approach. Furthermore, we show that our model, which is trained only on monolingual corpora, outperforms the standard statistical MT approach to learning phrase tables when trained on the same amount of sentence-aligned parallel corpora, when evaluated on Top 10 accuracy.

Spanish	English	Sim Score	Is present in lexicon
señores	gentlemen	0.99	NO
chipre	cyprus	0.66	YES
mujeres	women	0.65	YES
alemania	germany	0.65	YES
hombres	men	0.62	YES
expresar	express	0.60	YES
racismo	racism	0.59	YES
interior	internal	0.55	YES
gobierno	government	0.52	YES
francia	france	0.52	YES
cultural	cultural	0.51	YES
suecia	sweden	0.50	YES
fundamento	basis	0.48	YES
francesa	french	0.48	YES
entre	between	0.47	YES
origen	origin	0.46	YES
tráfico	traffic	0.45	YES
de	of	0.44	YES
social	social	0.43	YES
ruego	thank	0.43	NO

Table 6: List of 20 most confident mappings using the dependency context with the part-of-speech mapping model translating all word-types. Note that although the second best mapping in Table4 for noun-translation is for xenofobia with score 0.87, xenofobia is not among the 1000 most frequent words (of all word-types) and thus is not in this test set.

As a second contribution, we go beyond previous literature which evaluated only on nouns. We showed how preserving a word's part-of-speech in translation can improve performance. We further proposed a solution to an interesting sub-problem encountered on the way. Since part-of-speeech tagsets are not identical across two languages, we propose a way of learning their mapping automatically. Restricting candidate space based on this learned tagset mapping resulted in 18% improvement over the direct application of context-based model to all word-types.

Dependency trees help improve the context for translation substantially and their use opens up the question of how the context can be enriched further making use of the hidden structure that may provide clues for a word's translation. We also believe that the problem of learning the mapping between tagsets in two different languages can be used in general for other NLP tasks making use of projection of words and its morphological/syntactic properties between languages.

References

S. Buchholz and E. Marsi. 2006. Conll-X shared task on multilingual dependency parsing. *Proceedings of CoNLL*, pages 189–210.

Y. Cao and H. Li. 2002. Base Noun Phrase translation using web data and the EM algorithm. *Proceedings of COLING-Volume 1*, pages 1–7.

D. Chiang. 2007. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33(2):201–228.

P. Fung and L.Y. Yee. 1998. An IR Approach for Translating New Words from Nonparallel, Comparable Texts. *Proceedings of ACL*, 36:414–420.

A. Haghighi, P. Liang, T. Berg-Kirkpatrick, and D. Klein. 2008. Learning bilingual lexicons from monolingual corpora. *Proceedings of ACL-HLT*, pages 771–779.

Z. Harris. 1985. Distributional structure. *Katz, J. J. (ed.), The Philosophy of Linguistics*, pages 26–47.

P. Koehn and K. Knight. 2002. Learning a translation lexicon from monolingual corpora. *Proceedings of ACL Workshop on Unsupervised Lexical Acquisition*, pages 9–16.

P. Koehn, F.J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL-HLT*, pages 48–54.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. *Proceedings of ACL, companian volume*, pages 177–180.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. *MT Summit X*.

D. Lin and P. Pantel. 2002. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(04):343–360.

G.S. Mann and D. Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. *Proceedings of NAACL*, pages 151–158.

M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19(2):313–330.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. *Proceedings of EMNLP-HLT*, pages 523–530.

J. Nivre, J. Hall, S. Kubler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The conll 2007 shared task on dependency parsing. *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 915–932.

S. Pado and M. Lapata. 2007. Dependency-Based Construction of Semantic Space Models. *Computational Linguistics*, 33(2):161–199.

R. Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. *Proceedings of ACL*, pages 519–526.

C. Schafer and D. Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. *Proceedings of COLING*, pages 1–7.

An Intrinsic Stopping Criterion for Committee-Based Active Learning

Fredrik Olsson
SICS
Box 1263
SE-164 29 Kista, Sweden
fredrik.olsson@sics.se

Katrin Tomanek
Jena University Language & Information Engineering Lab
Friedrich-Schiller-Universität Jena
Fürstengraben 30, D-07743 Jena, Germany
katrin.tomanek@uni-jena.de

Abstract

As supervised machine learning methods are increasingly used in language technology, the need for high-quality annotated language data becomes imminent. Active learning (AL) is a means to alleviate the burden of annotation. This paper addresses the problem of knowing when to stop the AL process without having the human annotator make an explicit decision on the matter. We propose and evaluate an intrinsic criterion for committee-based AL of named entity recognizers.

1 Introduction

With the increasing popularity of supervised machine learning methods in language processing, the need for high-quality labeled text becomes imminent. On the one hand, the amount of readily available texts is huge, while on the other hand the labeling and creation of corpora based on such texts is tedious, error prone and expensive.

Active learning (AL) is one way of approaching the challenge of classifier creation and data annotation. Examples of AL used in language engineering include named entity recognition (Shen et al., 2004; Tomanek et al., 2007), text categorization (Lewis and Gale, 1994; Hoi et al., 2006), part-of-speech tagging (Ringger et al., 2007), and parsing (Thompson et al., 1999; Becker and Osborne, 2005).

AL is a supervised machine learning technique in which the learner is in control of the data used for learning – the control is used to query an oracle, typically a human, for the correct label of the unlabeled training instances for which the classifier learned so far makes unreliable predictions.

The AL process takes as input a set of labeled instances and a larger set of unlabeled instances, and produces a classifier and a relatively small set of newly labeled data. The overall goal is to obtain as good a classifier as possible, without having to mark-up and supply the learner with more than necessary data. The learning process aims at keeping the human annotation effort to a minimum, only asking for advice where the training utility of the result of such a query is high.

The approaches taken to AL in this paper are based on committees of classifiers with access to pools of data. Figure 1 outlines a prototypical committee-based AL loop. In this paper we focus on the question when AL-driven annotation should be stopped (Item 7 in Figure 1).

Usually, the progress of AL is illustrated by means of a learning curve which depicts how the classifier's performance changes as a result of increasingly more labeled training data being available. A learning curve might be used to address the issue of knowing when to stop the learning process – once the curve has leveled out, that is, when additional training data does not contribute (much) to increase the performance of the classifier, the AL process may be terminated. While in a random selection scenario, classifier performance can be estimated by cross-validation on the labeled data, AL requires a held-out annotated reference corpus. In AL, the performance of the classifier cannot be reliably estimated using the data labeled in the process since sampling strategies for estimating performance assume independently and identically distributed examples (Schütze et al., 2006). The whole point in AL is to obtain a distribution of instances that is skewed in favor of the base learner used.

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 138–146,
Boulder, Colorado, June 2009.

1. Initialize the process by applying *EnsembleGeneration-Method* using base learner B on labeled training data set D_L to obtain a committee of classifiers C.
2. Have each classifier in C predict a label for every instance in the unlabeled data set D_U, obtain labeled set $D_U{}'$.
3. From $D_U{}'$, select the most informative n instances to learn from, obtaining $D_U{}''$.
4. Ask the teacher for classifications of the instances I in $D_U{}''$.
5. Move I, with supplied classifications, from D_U to D_L.
6. Re-train using *EnsembleGenerationMethod* and base learner B on the newly extended D_L to obtain a new committee, C.
7. Repeat steps 2 through 6 until D_U is empty or some stopping criterion is met.
8. Output classifier learned using *EnsembleGeneration-Method* and base learner B on D_L.

Figure 1: A prototypical query by committee algorithm.

In practice, however, an annotated reference corpus is rarely available and its creation would be inconsistent with the goal of creating a classifier with as little human effort as possible. Thus, other ways of deciding when to stop AL are needed. In this paper, we propose an intrinsic stopping-criterion for committee-based AL of named entity recognizers. It is intrinsic in that it relies on the characteristics of the data and the base learner[1] rather than on external parameters, i.e., the stopping criterion does not require any pre-defined thresholds.

The paper is structured as follows. Section 2 sketches interpretations of ideal stopping points and describes the idea behind our stopping criterion. Section 3 outlines related work. Section 4 describes the experiments we have conducted concerning a named entity recognition scenario, while Section 5 presents the results which are then discussed in Section 6. Section 7 concludes the paper.

2 A stopping criterion for active learning

What is the ideal stopping point for AL? Obviously, annotation should be stopped at the latest when the *best* classifier for a scenario is yielded. However, depending on the scenario at hand, the "best" classifier could have different interpretations. In many papers on AL and stopping criteria, the best (or optimal) classifier is the one that yields the highest performance on a test set. It is assumed that AL-based annotation should be stopped as soon as this performance is reached. This could be generalized as stopping criteria based on maximal classifier performance. In practice, the trade-off between annotation effort and classifier performance is related to the achievable performance given the learner configuration and data under scrutiny: For instance, would we invest many hours of additional annotation effort just to possibly increase the classifier performance by a fraction of a percent? In this context, a stopping criterion may be based on classifier performance convergence, and consequently, we can define the best possible classifier to be one which cannot learn more from the remaining pool of data.

The intrinsic stopping criterion (ISC) we propose here focuses on the latter aspect of the ideal stopping point described above – exhaustiveness of the AL pool. We suggest to stop the annotation process of the data from a given pool when the base learner cannot learn (much) more from it. The definition of our intrinsic stopping criterion for committee-based AL builds on the notions of Selection Agreement (Tomanek et al., 2007), and Validation Set Agreement (Tomanek and Hahn, 2008).

The Selection Agreement (SA) is the agreement among the members of a decision committee regarding the classification of the *most informative* instance selected from the pool of unlabeled data in each AL round. The intuition underlying the SA is that the committee will agree more on the hard instances selected from the remaining set of unlabeled data as the AL process proceeds. When the members of the committee are in complete agreement, AL *should* be aborted since it no longer contributes to the overall learning process – in this case, AL is but a computationally expensive counterpart of random sampling. However, as pointed out by Tomanek et al. (2007), the SA hardly ever signals complete agreement and can thus not be used as the sole indicator of AL having reached the point at which it should be aborted.

The Validation Set Agreement (VSA) is the agree-

[1]The term *base learner (configuration)* refers to the combination of base learner, parameter settings, and data representation.

ment among the members of the decision committee concerning the classification of a held-out, unannotated data set (the validation set). The validation set stays the same throughout the entire AL process. Thus, the VSA is mainly affected by the performance of the committee, which in turn, is grounded in the information contained in the most informative instances in the pool of unlabeled data. Tomanek and colleagues argue that the VSA is thus a good approximation of the (progression of the) learning curve and can be employed as decision support for knowing when to stop annotating – from the slope of the VSA curve one can read whether further annotation will result in increased classifier performance.

We combine the SA and the VSA into a single stopping criterion by relating the agreement of the committee on a held-out validation set with that on the (remaining) pool of unlabeled data. If the SA is larger than the VSA, it is a signal that the decision committee is more in agreement concerning the most informative instances in the (diminishing) unlabeled pool than it is concerning the validation set. This, in turn, implies that the committee would learn more from a random sample[2] from the validation set (or from a data source exhibiting the same distribution of instances), than it would from the unlabeled data pool. Based on this argument, a stopping criterion for committee-based AL can be formulated as:

> Active learning may be terminated when the Selection Agreement is larger than, or equal to, the Validation Set Agreement.

In relation to the stopping criterion based solely on SA proposed by Tomanek et al. (2007), the above defined criterion comes into effect earlier in the AL process. Furthermore, while it was claimed in (Tomanek and Hahn, 2008) that one can observe the classifier convergence from the VSA curve (as it approximated the progression of the learning curve), that requires a threshold to be specified for the actual stopping point. The ISC is completely intrinsic and does thus not require any thresholds to be set.

3 Related work

Schohn and Cohn (2000) report on document classification using AL with Support Vector Machines. If the most informative instance is no closer to the decision hyperplane than any of the support vectors, the margin has been exhausted and AL is terminated.

Vlachos (2008) suggests to use classifier confidence to define a stopping criterion for uncertainty-based sampling. The idea is to stop learning when the confidence of the classifier, on an external, possibly unannotated test set, remains at the same level or drops for a number of consecutive iterations during the AL process. Vlachos shows that the criterion indeed is applicable to the tasks he investigates.

Zhu and colleagues (Zhu and Hovy, 2007; Zhu et al., 2008a; Zhu et al., 2008b) introduce *max-confidence*, *min-error*, *minimum expected error strategy*, *overall-uncertainty*, and *classification-change* as means to terminate AL. They primarily use a single-classifier approach to word sense disambiguation and text classification in their experiments. *Max-confidence* seeks to terminate AL once the classifier is most confident in its predictions. In the *min-error* strategy, the learning is halted when there is no difference between the classifier's predictions and those labels provided by a human annotator. The *minimum expected error strategy* involves estimating the classification error on future unlabeled instances and stop the learning when the expected error is as low as possible. *Overall-uncertainty* is similar to max-confidence, but unlike the latter, overall-uncertainty takes into account all data remaining in the unlabeled pool when estimating the uncertainty of the classifier. *Classification-change* builds on the assumption that the most informative instance is the one which causes the classifier to change the predicted label of the instance. Classification-change-based stopping is realized by Zhu and colleagues such that AL is terminated once no predicted label of the instances in the unlabeled pool change during two consecutive AL iterations.

Laws and Schütze (2008) investigate three ways of terminating uncertainty-based AL for named entity recognition – *minimal absolute performance*, *maximum possible performance*, and *convergence*. The *minimal absolute performance* of the system is set by the user prior to starting the AL process. The classifier then estimates its own performance using a held-out unlabeled data set. Once the performance is reached, the learning is terminated. The *maximum possible performance* strategy refers to

[2] The sample has to be large enough to mimic the distribution of instances in the original unlabeled pool.

the optimal performance of the classifier given the data. Once the optimal performance is achieved, the process is aborted. Finally, the *convergence* criterion aims to stop the learning process when the pool of available data does not contribute to the classifier's performance. The convergence is calculated as the gradient of the classifier's estimated performance or uncertainty. Laws and Schütze conclude that both gradient-based approaches, that is, convergence, can be used as stopping criteria relative to the optimal performance achievable on a given pool of data. They also show that while their method lends itself to acceptable estimates of accuracy, it is much harder to estimate the recall of the classifier. Thus, the stopping criteria based on minimal absolute or maximum possible performance are not reliable.

The work most related to ours is that of Tomanek and colleagues (Tomanek et al., 2007; Tomanek and Hahn, 2008) who define and evaluate the *Selection Agreement* (SA) and the *Validation Set Agreement* (VSA) already introduced in Section 2. Tomanek and Hahn (2008) conclude that monitoring the progress of AL should be based on a separate validation set instead of the data directly affected by the learning process – thus, VSA is preferred over SA. Further, they find that the VSA curve approximates the progression of the learning curve and thus classifier performance convergence could be estimated. However, to actually find where to stop the annotation, a threshold needs to be set.

Our proposed intrinsic stopping criterion is unique in several ways: The ISC is intrinsic, relying only on the characteristics of the base learner and the data at hand in order to decide when the AL process may be terminated. The ISC does not require the user to set any external parameters prior to initiating the AL process. Further, the ISC is designed to work with committees of classifiers, and as such, it is independent of how the disagreement between the committee members is quantified. The ISC does neither rely on a particular base learner, nor on a particular way of creating the decision committee.

4 Experiments

To challenge the definition of the ISC, we conducted two types of experiments concerning named entity recognition. The primary focus of the first type of experiment is on creating classifiers (classifier-centric), while the second type is concerned with the creation of annotated documents (data-centric). In all experiments, the agreement among the decision committee members is quantified by the Vote Entropy measure (Engelson and Dagan, 1996):

$$VE(e) = -\frac{1}{\log k} \sum_{l} \frac{V(l,e)}{k} \log \frac{V(l,e)}{k} \qquad (1)$$

where k is the number of members in the committee, and $V(l,e)$ is the number of members assigning label l to instance e. If an instance obtains a low Vote Entropy value, it means that the committee members are in high agreement concerning its classification, and thus also that it is less a informative one.

4.1 Classifier-centric experimental settings

In common AL scenarios, the main goal of using AL is to create a good classifier with minimal label complexity. To follow this idea, we select sentences that are assumed to be useful for classifier training. We decided to select complete sentences – instead of, e.g., single tokens – as in practice annotators must see the context of words to decide on their entity labels.

Our experimental setting is based on the AL approach described by Tomanek et al. (2007): The committee consists of $k = 3$ Maximum Entropy (ME) classifiers (Berger et al., 1996). In each AL iteration, each classifier is trained on a randomly drawn (sampling without replacement) subset $L' \subset L$ with $|L'| = \frac{2}{3}L$, L being the set of all instances labeled so far (cf. *EnsembleGenerationMethod* in Figure 1). Usefulness of a sentence is estimated as the average token Vote Entropy (cf. Equation 1). In each AL iteration, the 20 most useful sentences are selected ($n = 20$ in Step 3 in Figure 1). AL is started from a randomly chosen seed of 20 sentences.

While we made use of ME classifiers during the selection, we employed an NE tagger based on Conditional Random Fields (CRF) (Lafferty et al., 2001) during evaluation time to determine the learning curves. CRFs have a significantly higher tagging performance, so the final classifier we are aiming at should be a CRF model. We have shown before (Tomanek et al., 2007) that MEs are well apt as selectors with the advantage of much shorter training times than CRFs. For both MEs and CRFs the

same features were employed which comprised orthographical (based mainly on regular expressions), lexical and morphological (suffixed/prefixed, word itself), syntactic (POS tags), as well as contextual (features of neighboring tokens) ones.

The experiments on classifier-centric AL have been performed on the English data set of corpus used in the CoNLL-2003 shared task (Tjong Kim Sang and Meulder, 2003). This corpus consists of newspaper articles annotated with respect to person, location, and organisation entities. As AL pool we took the training set which consists of about 14,000 sentences ($\approx 200,000$ tokens). As validation set and as gold standard for plotting the learning curve we used CoNLL's evaluation corpus which sums up to 3,453 sentences.

4.2 Data-centric experimental settings

While AL is commonly used to create as good classifiers as possible, with the amount of human effort kept to a minimum, it may result in fragmented and possibly non re-usable annotations (e.g., a collection of documents in which only some of the names are marked up). This experiment concerns a method of orchestrating AL in a way beneficial for the bootstrapping of annotated data (Olsson, 2008). The bootstrapping proper is realized by means of AL for selecting *documents* to annotate, as opposed to *sentences*. This way the annotated data set is comprised of entire documents thus promoting data creation. As in the classifier-centric setting, the task is to recognize names – persons, organizations, locations, times, dates, monetary expressions, and percentages – in news wire texts. The texts used are part of the MUC-7 corpus (Linguistic Data Consortium, 2001) and consists of 100 documents, 3,480 sentences, and 90,790 tokens. The task is approached using the IOB tagging scheme proposed by, e.g., Ramshaw and Marcus (1995), turning the original 7-class task into a 15-class task. Each token is represented using a fairly standard menagerie of features, including such stemming from the surface appearance of the token (e.g., *Contains dollar? Length in characters*), calculated based on linguistic pre-processing made with the English Functional Dependency Grammar (Tapanainen and Järvinen, 1997) (e.g., *Case, Part-of-speech*), fetched from precompiled lists of information (e.g., *Is first name?*), and features based on predictions concerning the context of the token (e.g, *Class of previous token*).

The decision committee is made up from 10 boosted decision trees using MultiBoostAB (Webb, 2000) (cf. *EnsembleGenerationMethod* in Figure 1). Each classifier is created by the REPTree decision tree learner described by Witten and Frank (2005). The informativeness of a document is calculated by means of average token Vote Entropy (cf. Equation 1). The seed set of the AL process consists of five randomly selected documents. In each AL iteration, one document is selected for annotation from the corpus ($n = 1$ in Step 3 in Figure 1).

5 Results

Two different scenarios were used to illustrate the applicability of the proposed intrinsic stopping criterion. In the first scenario, we assumed that the pool of unlabeled data was static and fairly large. In the second scenario, we assumed that the unlabeled data would be collected in smaller batches as it was made available on a stream, for instance, from a news feed. Both the classifier-centric and the data-centric experiments were carried out within the first scenario. Only the classifier-centric experiment was conducted in the stream-based scenario.

In the classifier-centric setting, the SA is defined as (1 − Vote Entropy) for the most informative instances in the unlabeled pool, that is, the per-token average Vote Entropy on the most informative sentences. Analogously, in the data-centric setting, the SA is defined as (1 − Vote Entropy) for the most informative document – here too, the informativeness is calculated as the per-token average Vote Entropy. In both settings, the VSA is the per-token average Vote Entropy on the validation set.

5.1 AL on static pools

The intersection of the SA and VSA agreement curves indicates a point at which the AL process may be terminated without (a significant) loss in classifier performance. For both AL scenarios (data- and classifier-centric) we plot both the learning curves for AL and random selection, as well as the SA and VSA curve for AL. In both scenarios, these

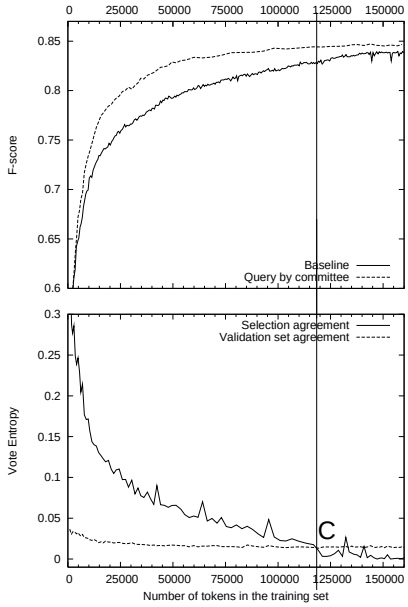


Figure 2: Classifier-centric AL experiments on the CoNLL corpus. The intersection, *C*, corresponds to the point where (almost) no further improvement in terms of classifier performance can be expected. The baseline learning curve shows the results of learning from randomly sampled data.

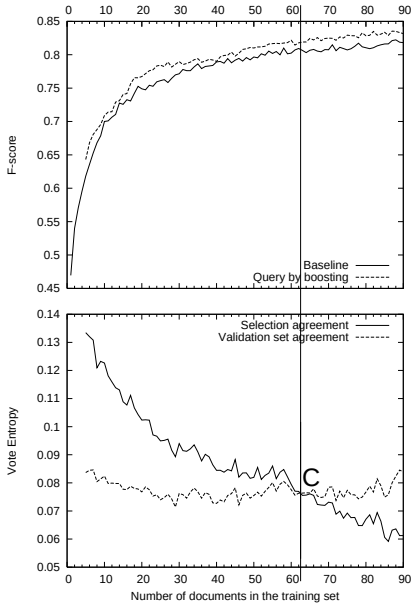


Figure 3: Data-centric AL experiments on the MUC-7 corpus. The intersection, *C*, corresponds to a point at which the AL curve has almost leveled out. The baseline learning curve shows the results of learning from randomly sampled data.

curves are averages over several runs.[3]

The results from the classifier-centric experiment on the CoNLL corpus are presented in Figure 2. AL clearly outperforms random selection. The AL curve converges at a maximum performance of $F \approx 84\%$ after about 125,000 tokens. As expected, the SA curve drops from high values in the beginning down to very low values in the end where hardly any interesting instances are left in the pool. The intersection (*C*) with the VSA curve is very close to the point (125,000 tokens) where no further increase of performance can be reached by additional annotation making it a good stopping point.

The results from the data-centric experiment are available in Figure 3. The bottom part shows the SA and VSA curves. The ISC occurs at the intersection of the SA and VSA curves (*C*), which corresponds to a point well beyond the steepest part of the learning curve. While stopping the learning at *C* results in a classifier with performance inferior what is maximally achievable, stopping at *C* arguably corresponds to a plausible place to abort the learning. The optimal performance is $F \approx 83.5\%$, while the ISC corresponds to $F \approx 82\%$.

Keep in mind that the learning curves with which the ISC are compared are not available in a practical situation, they are included in Figures 2 and 3 for the sake of clarity only.

5.2 AL on streamed data

One way of paraphrasing the ISC is: Once the intersection between the SA and VSA curves has been reached, the most informative instances remaining in the pool of unlabeled data are less informative to the classifier than the instances in the held-out, unlabeled validation set are on average. This means that the classifier would learn more from a sufficiently large sample taken from the validation set than it would if the AL process continued on the remaining unlabeled pool.[4]

As an illustration of the practical applicability of the ISC consider the following scenario. Assume

[3]The classifier-centric experiments are averages over three independent runs. The data-centric experiments are averages over ten independent runs.

[4]Note however, that the classifier might still learn from the instances in the unlabeled pool – applying the ISC only means that the classifier would learn more from a validation set-like distribution of instances.

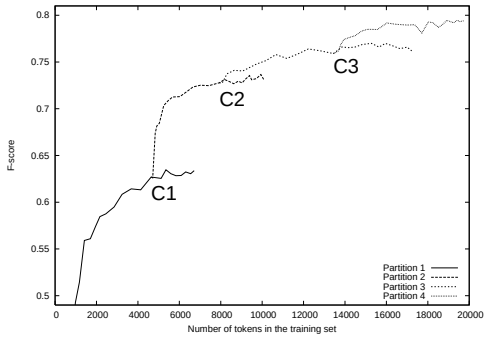


Figure 4: AL curves for the four partitions used in the experiments on streamed data. C_i denotes a point at which the AL is terminated for partition i and a new partition is employed instead. *C1* corresponds to the ISC plotted in the graph labeled *Partition 1* in Figure 5, *C2* to the ISC in *Partition 2*, and *C3* to the ISC in *Partition 3*.

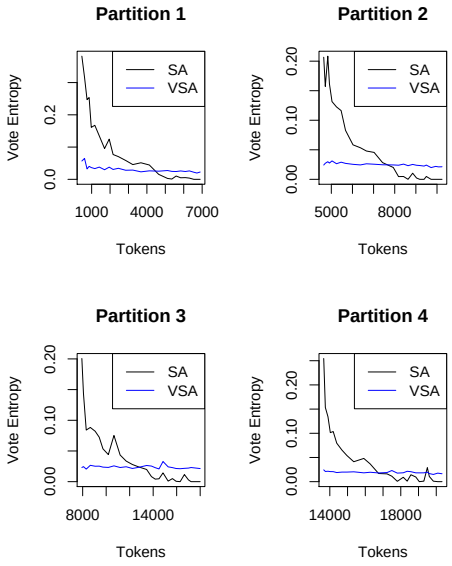


Figure 5: The SA and VSA curves for the four data partitions used in the experiment on streamed data. Each intersection – ISC – corresponds to a point where AL is terminated.

that we are collecting data from a stream, for instance items taken from a news feed. Thus, the data is not available on the form of a closed set, but rather an open one which grows over time. To make the most of the human annotators in this scenario, we want them to operate on batches of data instead of annotating individual news items as they are published. The purpose of the annotation is to mark up names in the texts in order to train a named entity recognizer. To do so, we wait until there has appeared a given number of sentences on the stream, and then collect those sentences. The problem is, how do we know when the AL-based annotation process for each such batch should be terminated? We clearly do not want the annotators to annotate all sentences, and we cannot have the annotators set new thresholds pertaining to the absolute performance of the named entity recognizer for each new batch of data available. By using the ISC, we are able to automatically issue a halting of the AL process (and thus also the annotation process) and proceed to the next batch of data without losing too much in performance, and without having the annotators mark up too much of the available data. To this end, the ISC seems like a reasonable trade-off between annotation effort and performance gain.

To carry out this experiment we took a sub sample of 10% (1,400 sentences) from the original AL pool of the CoNLL corpus as validation set.[5] The rest of this pool was split into batches of about 500 consecutive sentences. Classifier-centric AL was now run taking the first batch as pool to select from. At the point where the SA and VSA curve crossed, we continued AL selection from the next batch and so forth. Figure 4 shows the learning curve for a simulation of the scenario described above. The intersection between the SA and VSA curves for partition 1 as depicted in Figure 5 corresponds to the first "step" (ending in *C1*) in the stair-like learning curve in Figure 4. The step occurs after 4,641 tokens. Analogously, the other steps (ending in *C2* and *C3*, respectively) in the learning curve corresponds the intersection between the SA and VSA curves for partitions 2 and 3 in Figure 5. The intersection for partition 4 corresponds to the point were we would have turned to the next partition. This experiment was stopped after 4 partitions.

Table 1 shows the accumulated number of sentences and tokens (center columns) that required annotation in order to reach the ISC for each partition. In addition, the last column in the table shows the number of sentences (of the 500 collected for inclu-

[5]Note that the original CoNLL test set was not used in this experiment, thus the F-score reported in Figure 4 cannot be compared to that in Figure 2.

Partition	Sents	Toks	Sentences per partition
1	320	4,641	320
2	580	7,932	260
3	840	13,444	260
4	1070	16,751	230

Table 1: The number of tokens and sentences required to reach the ISC for each partition.

sion in each partition) needed to reach the ISC – each new partition contributes less to the increase in performance than the preceding ones.

6 Discussion

We have argued that one interpretation of the ISC is that it constitutes the point where the informativeness on the remaining part of the AL pool is lower than the informativeness on a different and independent data set with the same distribution. In the first AL scenario where there is one static pool to select from, reaching this point can be interpreted as an overall stopping point for annotation. Here, the ISC represents a trade-off between the amount of data annotated and the classifier performance obtained such that the resulting classifier is nearly optimal with respect to the data at hand. In the second, stream-based AL scenario where several smaller partitions are consecutively made available to the learner, the ISC serves as an indicator that the annotation of one batch should be terminated, and that the mark-up should proceed with the next batch.

The ISC constitutes an intrinsic way of determining when to stop the learning process. It does not require any external parameters such as pre-defined thresholds to be set, and it depends only on the characteristics of the data and base learner at hand. The ISC can be utilized to relate the performance of the classifier to the performance that is possible to obtain by the data and learner at hand.

The ISC can not be used to estimate the performance of the classifier. Consequently, it can not be used to relate the classifier's performance to an externally set level, such as a particular F-score provided by the user. In this sense, the ISC may serve as a complement to stopping criteria requiring the classifier to achieve absolute performance measures before the learning process is aborted, for instance the *max-confidence* proposed by Zhu and Hovy (2007), and the *minimal absolute performance* introduced by Laws and Schütze (2008).

7 Conclusions and Future Work

We have defined and empirically tested an intrinsic stopping criterion (ISC) for committee-based AL. The results of our experiments in two named entity recognition scenarios show that the stopping criterion is indeed a viable one, which represents a fair trade-off between data use and classifier performance. In a setting in which the unlabeled pool of data used for learning is static, terminating the learning process by means of the ISC results in a nearly optimal classifier. The ISC can also be used for deciding when the pool of unlabeled data needs to be refreshed.

We have focused on challenging the ISC with respect to named entity recognition, approached in two very different settings; future work includes experiments using the ISC for other tasks. We believe that the ISC is likely to work in AL-based approaches to, e.g., part-of-speech tagging, and chunking as well. It should be kept in mind that while the types of experiments conducted here concern the same task, the ways they are realized differ in many respects: the ways the decision committees are formed, the data sets used, the representation of instances, the relation between the sample size and the instance size, as well as the pre-processing tools used. Despite these differences, which outnumbers the similarities, the ISC proves a viable stopping criterion.

An assumption underlying the ISC is that the initial distribution of instances in the pool of unlabeled data used for learning, and the distribution of instances in the validation set are the same (or at least very similar). Future work also includes investigations of automatic ways to ensure that this assumption is met.

Acknowledgements

The first author was funded by the EC project COMPANIONS (IST-FP6-034434), the second author was funded by the EC projects BOOTStrep (FP6-028099) and CALBC (FP7-231727).

References

Markus Becker and Miles Osborne. 2005. A Two-Stage Method for Active Learning of Statistical Grammars. In *Proc 19th IJCAI*, Edinburgh, Scotland, UK.

Adam Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

Sean P. Engelson and Ido Dagan. 1996. Minimizing Manual Annotation Cost In Supervised Training From Corpora. In *Proc 34th ACL*, Santa Cruz, California, USA.

Steven C. H. Hoi, Rong Jin, and Michael R. Lyu. 2006. Large-Scale Text Categorization by Batch Mode Active Learning. In *Proc 15th WWW*, Edinburgh, Scotland.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc 18th ICML*, Williamstown, Massachusetts, USA.

Florian Laws and Hinrich Schütze. 2008. Stopping Criteria for Active Learning of Named Entity Recognition. In *Proc 22nd COLING*, Manchester, England.

David D. Lewis and William A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proc 17th ACM-SIGIR*, Dublin, Ireland.

Linguistic Data Consortium. 2001. Message understanding conference (muc) 7. LDC2001T02. FTP FILE. Philadelphia: Linguistic Data Consortium.

Fredrik Olsson. 2008. *Bootstrapping Named Entity Annotation by means of Active Machine Learning – A Method for Creating Corpora*. Ph.D. thesis, Department of Swedish, University of Gothenburg.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text Chunking using Transformation Based Learning. In *Proc 3rd VLC*, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi, and Deryle Lonsdale. 2007. Active Learning for Part-of-Speech Tagging: Accelerating Corpus Annotation. In *Proc Linguistic Annotation Workshop*, Prague, Czech Republic.

Greg Schohn and David Cohn. 2000. Less is More: Active Learning with Support Vector Machines. In *Proc 17th ICML*, Stanford University, Stanford, California, USA.

Hinrich Schütze, Emre Velipasaoglu, and Jan O. Pedersen. 2006. Performance Thresholding in Practical Text Classification. In *Proc 15th CIKM*, Arlington, Virginia, USA.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-Criteria-based Active Learning for Named Entity Recognition. In *Proc 42nd ACL*, Barcelona, Spain.

Pasi Tapanainen and Timo Järvinen. 1997. A Non-Projective Dependency Parser. In *Proc 5th ANLP*, Washington DC, USA.

Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. 1999. Active Learning for Natural Language Parsing and Information Extraction. In *Proc 16th ICML*, Bled, Slovenia.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language Independent Named Entity Recognition. In *Proc 7th CoNLL*, Edmonton, Alberta, Canada.

Katrin Tomanek and Udo Hahn. 2008. Approximating Learning Curves for Active-Learning-Driven Annotation. In *Proc 6th LREC*, Marrakech, Morocco.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. An Approach to Text Corpus Construction which Cuts Annotation Costs and Maintains Reusability of Annotated Data. In *Proc Joint EMNLP-CoNLL*, Prague, Czech Republic.

Andreas Vlachos. 2008. A Stopping Criterion for Active Learning. *Computer, Speech and Language*, 22(3):295–312, July.

Geoffrey I. Webb. 2000. MultiBoosting: A Technique for Combining Boosting and Wagging. *Machine Learning*, 40(2):159–196, August.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools with Java Implementations. 2nd Edition.* Morgan Kaufmann, San Fransisco.

Jingbo Zhu and Eduard Hovy. 2007. Active Learning for Word Sense Disambiguation with Methods for Addressing the Class Imbalance Problem. In *Proc Joint EMNLP-CoNLL*, Prague, Czech Republic.

Jingbo Zhu, Huizhen Wang, and Eduard Hovy. 2008a. Learning a Stopping Criterion for Active Learning for Word Sense Disambiguation and Text Classification. In *Proc 3rd IJCNLP*, Hyderabad, India.

Jingbo Zhu, Huizhen Wang, and Eduard Hovy. 2008b. Multi-Criteria-based Strategy to Stop Active Learning for Data Annotation. In *Proc 22nd COLING*, Manchester, England.

Design Challenges and Misconceptions in Named Entity Recognition*†‡

Lev Ratinov **Dan Roth**
Computer Science Department
University of Illinois
Urbana, IL 61801 USA
{ratinov2,danr}@uiuc.edu

Abstract

We analyze some of the fundamental design challenges and misconceptions that underlie the development of an efficient and robust NER system. In particular, we address issues such as the representation of text chunks, the inference approach needed to combine local NER decisions, the sources of prior knowledge and how to use them within an NER system. In the process of comparing several solutions to these challenges we reach some surprising conclusions, as well as develop an NER system that achieves 90.8 F_1 score on the CoNLL-2003 NER shared task, the best reported result for this dataset.

1 Introduction

Natural Language Processing applications are characterized by making complex interdependent decisions that require large amounts of prior knowledge. In this paper we investigate one such application– Named Entity Recognition (NER). Figure 1 illustrates the necessity of using prior knowledge and non-local decisions in NER. In the absence of mixed case information it is difficult to understand that *"BLINKER"* is a person. Likewise, it is not obvious that the last mention of *"Wednesday"* is an organization (in fact, the first mention of *"Wednesday"* can also be understood as a "comeback" which happens on Wednesday). An NER system could take advantage of the fact that "blinker" is also mentioned later in the text as the easily identifiable "Reggie Blinker". It is also useful to know that *Udinese* is a soccer club (an entry about this club appears in Wikipedia), and the expression *"both Wednesday and Udinese"* implies that *"Wednesday"* and *"Udinese"* should be assigned the same label.

SOCCER - [PER BLINKER] BAN LIFTED .
[LOC LONDON] 1996-12-06 [MISC Dutch] forward [PER Reggie Blinker] had his indefinite suspension lifted by [ORG FIFA] on Friday and was set to make his [ORG Sheffield Wednesday] comeback against [ORG Liverpool] on Saturday . [PER Blinker] missed his club's last two games after [ORG FIFA] slapped a worldwide ban on him for appearing to sign contracts for both [ORG Wednesday] and [ORG Udinese] while he was playing for [ORG Feyenoord].

Figure 1: Example illustrating challenges in NER.

The above discussion focuses on the need for external knowledge resources (for example, that *Udinese* can be a soccer club) and the need for non-local features to leverage the multiple occurrences of named entities in the text. While these two needs have motivated some of the research in NER in the last decade, several other fundamental decisions must be made. These include: what model to use for

* The system and the Webpages dataset are available at: http://l2r.cs.uiuc.edu/∼cogcomp/software.php

† This work was supported by NSF grant NSF SoD-HCER-0613885, by MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC and by an NDIIPP project from the National Library of Congress.

‡ We thank Nicholas Rizzolo for the baseline LBJ NER system, Xavier Carreras for suggesting the word class models, and multiple reviewers for insightful comments.

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 147–155,
Boulder, Colorado, June 2009.

sequential inference, how to represent text chunks and what inference (decoding) algorithm to use.

Despite the recent progress in NER, the effort has been dispersed in several directions and there are no published attempts to compare or combine the recent advances, leading to some design misconceptions and less than optimal performance. In this paper we analyze some of the fundamental design challenges and misconceptions that underlie the development of an efficient and robust NER system. We find that BILOU representation of text chunks significantly outperforms the widely adopted BIO. Surprisingly, naive greedy inference performs comparably to beamsearch or Viterbi, while being considerably more computationally efficient. We analyze several approaches for modeling non-local dependencies proposed in the literature and find that none of them clearly outperforms the others across several datasets. However, as we show, these contributions are, to a large extent, independent and, as we show, the approaches can be used together to yield better results. Our experiments corroborate recently published results indicating that word class models learned on unlabeled text can significantly improve the performance of the system and can be an alternative to the traditional semi-supervised learning paradigm. Combining recent advances, we develop a publicly available NER system that achieves 90.8 F_1 score on the CoNLL-2003 NER shared task, the best reported result for this dataset. Our system is robust – it consistently outperforms all publicly available NER systems (e.g., the Stanford NER system) on all three datasets.

2 Datasets and Evaluation Methodology

NER system should be robust across multiple domains, as it is expected to be applied on a diverse set of documents: historical texts, news articles, patent applications, webpages etc. Therefore, we have considered three datasets: CoNLL03 shared task data, MUC7 data and a set of Webpages we have annotated manually. In the experiments throughout the paper, we test the ability of the tagger to adapt to new test domains. Throughout this work, we train on the CoNLL03 data and test on the other datasets *without retraining*. The differences in annotation schemes across datasets created evaluation challenges. We discuss the datasets and the evaluation methods below.

The CoNLL03 shared task data is a subset of Reuters 1996 news corpus annotated with 4 entity types: *PER,ORG, LOC, MISC*. It is important to notice that *both* the training and the development datasets are news feeds from *August* 1996, while the test set contains news feeds from *December* 1996. The named entities mentioned in the test dataset are considerably different from those that appear in the training or the development set. As a result, the test dataset is considerably harder than the development set. **Evaluation:** Following the convention, we report phrase-level F_1 score.

The MUC7 dataset is a subset of the North American News Text Corpora annotated with a wide variety of entities including people, locations, organizations, temporal events, monetary units, and so on. Since there was no direct mapping from temporal events, monetary units, and other entities from MUC7 and the MISC label in the CoNLL03 dataset, we measure performance only on *PER,ORG* and *LOC*. **Evaluation:** There are several sources of inconsistency in annotation between MUC7 and CoNLL03. For example, since the MUC7 dataset does not contain the *MISC* label, in the sentence *"balloon, called the Virgin Global Challenger"* , the expression *Virgin Global Challenger* should be labeled as *MISC* according to CoNLL03 guidelines. However, the gold annotation in MUC7 is *"balloon, called the* [*ORG Virgin*] *Global Challenger"*. These and other annotation inconsistencies have prompted us to relax the requirements of finding the exact phrase boundaries and measure performance using token-level F_1.

Webpages - we have assembled and manually annotated a collection of 20 webpages, including personal, academic and computer-science conference homepages. The dataset contains 783 entities (96-loc, 223-org, 276-per, 188-misc). **Evaluation:** The named entities in the webpages were highly ambiguous and very different from the named entities seen in the training data. For example, the data included sentences such as : *"Hear, O Israel, the Lord our God, the Lord is one."* We could not agree on whether *"O Israel"* should be labeled as *ORG, LOC,* or *PER*. Similarly, we could not agree on whether *"God"* and *"Lord"* is an *ORG* or *PER*. These issues

led us to report token-level entity-identification F_1 score for this dataset. That is, if a named entity token was identified as such, we counted it as a correct prediction ignoring the named entity type.

3 Design Challenges in NER

In this section we introduce the baseline NER system, and raise the fundamental questions underlying robust and efficient design. These questions define the outline of this paper. NER is typically viewed as a sequential prediction problem, the typical models include HMM (Rabiner, 1989), CRF (Lafferty et al., 2001), and sequential application of Perceptron or Winnow (Collins, 2002). That is, let $\mathbf{x} = (x_1, \ldots, x_N)$ be an input sequence and $\mathbf{y} = (y_1, \ldots, y_N)$ be the output sequence. The sequential prediction problem is to estimate the probabilities

$$P(y_i | x_{i-k} \ldots x_{i+l}, y_{i-m} \ldots y_{i-1}),$$

where k, l and m are small numbers to allow tractable inference and avoid overfitting. This conditional probability distribution is estimated in NER using the following baseline set of features (Zhang and Johnson, 2003): (1) previous two predictions y_{i-1} and y_{i-2} (2) current word x_i (3) x_i word type (all-capitalized, is-capitalized, all-digits, alphanumeric, etc.) (4) prefixes and suffixes of x_i (5) tokens in the window $c = (x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2})$ (6) capitalization pattern in the window c (7) conjunction of c and y_{i-1}.

Most NER systems use additional features, such as POS tags, shallow parsing information and gazetteers. We discuss additional features in the following sections. We note that we normalize dates and numbers, that is *12/3/2008* becomes **Date**, *1980* becomes **DDDD** and *212-325-4751* becomes **DDD*-*DDD*-*DDDD**. This allows a degree of abstraction to years, phone numbers, etc.

Our baseline NER system uses a regularized averaged perceptron (Freund and Schapire, 1999). Systems based on perceptron have been shown to be competitive in NER and text chunking (Kazama and Torisawa, 2007b; Punyakanok and Roth, 2001; Carreras et al., 2003) We specify the model and the features with the LBJ (Rizzolo and Roth, 2007) modeling language. We now state the four fundamental design decisions in NER system which define the structure of this paper.

Algorithm	Baseline system	Final System
Greedy	83.29	90.57
Beam size=10	83.38	90.67
Beam size=100	83.38	90.67
Viterbi	83.71	N/A

Table 1: Phrase-level F_1 performance of different inference methods on CoNLL03 test data. Viterbi cannot be used in the end system due to non-local features.

Key design decisions in an NER system.

1) How to represent text chunks in NER system?

2) What inference algorithm to use?

3) How to model non-local dependencies?

4) How to use external knowledge resources in NER?

4 Inference & Chunk Representation

In this section we compare the performance of several inference (decoding) algorithms: greedy left-to-right decoding, Viterbi and beamsearch. It may appear that beamsearch or Viterbi will perform much better than naive greedy left-to-right decoding, which can be seen as beamsearch of size one. The Viterbi algorithm has the limitation that it does not allow incorporating some of the non-local features which will be discussed later, therefore, we cannot use it in our end system. However, it has the appealing quality of finding the most likely assignment to a second-order model, and since the baseline features only have second order dependencies, we have tested it on the baseline configuration.

Table 1 compares between the greedy decoding, beamsearch with varying beam size, and Viterbi, both for the system with baseline features and for the end system (to be presented later). Surprisingly, the greedy policy performs well, this phenmenon was also observed in the POS tagging task (Toutanova et al., 2003; Roth and Zelenko, 1998). The implications are subtle. First, due to the second-order of the model, the greedy decoding is over 100 times faster than Viterbi. The reason is that with the BILOU encoding of four NE types, each token can take 21 states (*O, B-PER, I-PER , U-PER, etc.*). To tag a token, the greedy policy requires 21 comparisons, while the Viterbi requires 21^3, and this analysis carries over to the number of classifier invocations. Furthermore, both beamsearch and Viterbi require transforming the predictions of the classi-

Rep. Scheme	CoNLL03 Test	CoNLL03 Dev	MUC7 Dev	MUC7 Test
BIO	89.15	**93.61**	86.76	85.15
BILOU	**90.57**	93.28	**88.09**	**85.62**

Table 2: End system performance with BILOU and BIO schemes. BILOU outperforms the more widely used BIO.

fiers to probabilities as discussed in (Niculescu-Mizil and Caruana, 2005), incurring additional time overhead. Second, this result reinforces the intuition that global inference over the second-order HMM features does not capture the non-local properties of the task. The reason is that the NEs tend to be short chunks separated by multiple "outside" tokens. This separation "breaks" the Viterbi decision process to independent maximization of assignment over short chunks, where the greedy policy performs well. On the other hand, dependencies between isolated named entity chunks have *longer*-range dependencies and are not captured by second-order transition features, therefore requiring separate mechanisms, which we discuss in Section 5.

Another important question that has been studied extensively in the context of shallow parsing and was somewhat overlooked in the NER literature is the representation of text segments (Veenstra, 1999). Related works include voting between several representation schemes (Shen and Sarkar, 2005), lexicalizing the schemes (Molina and Pla, 2002) and automatically searching for best encoding (Edward, 2007). However, we are not aware of similar work in the NER settings. Due to space limitations, we do not discuss all the representation schemes and combining predictions by voting. We focus instead on two most popular schemes– BIO and BILOU. The BIO scheme suggests to learn classifiers that identify the **B**eginning, the **I**nside and the **O**utside of the text segments. The BILOU scheme suggests to learn classifiers that identify the **B**eginning, the **I**nside and the **L**ast tokens of multi-token chunks as well as **U**nit-length chunks. The BILOU scheme allows to learn a more expressive model with only a small increase in the number of parameters to be learned. Table 2 compares the end system's performance with BIO and BILOU. Examining the results, we reach two conclusions: (1) choice of encoding scheme has a big impact on the system performance and (2) the less used BILOU formalism significantly outperforms the widely adopted BIO tagging scheme. We use the BILOU scheme throughout the paper.

5 Non-Local Features

The key intuition behind non-local features in NER has been that identical tokens should have identical label assignments. The sample text discussed in the introduction shows one such example, where all occurrences of *"blinker"* are assigned the *PER* label. However, in general, this is not always the case; for example we might see in the same document the word sequences *"Australia"* and *"The bank of Australia"*. The first instance should be labeled as *LOC*, and the second as *ORG*. We consider three approaches proposed in the literature in the following sections. Before continuing the discussion, we note that we found that adjacent documents in the CoNLL03 and the MUC7 datasets often discuss the same entities. Therefore, we ignore document boundaries and analyze global dependencies in 200 and 1000 token windows. These constants were selected by hand after trying a small number of values. We believe that this approach will also make our system more robust in cases when the document boundaries are not given.

5.1 Context aggregation

(Chieu and Ng, 2003) used features that aggregate, for each document, the context tokens appear in. Sample features are: *the longest capitilized sequence of words in the document which contains the current token* and *the token appears before a company marker such as ltd. elsewhere in text*. In this work, we call this type of features *context aggregation features*. Manually designed context aggregation features clearly have low coverage, therefore we used the following approach. Recall that for each token instance x_i, we use as features the tokens in the window of size two around it: $c_i = (x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2})$. When the same token type t appears in several locations in the text, say $x_{i_1}, x_{i_2}, \ldots, x_{i_N}$, for each instance x_{i_j}, in addition to the context features c_{i_j}, we also aggregate the context across all instances within 200 tokens: $C = \cup_{j=1}^{j=N} c_{i_j}$.

	Component	CoNLL03 Test data	CoNLL03 Dev data	MUC7 Dev	MUC7 Test	Web pages
1)	Baseline	83.65	89.25	74.72	71.28	71.41
2)	(1) + Context Aggregation	85.40	89.99	**79.16**	71.53	70.76
3)	(1) + Extended Prediction History	**85.57**	**90.97**	78.56	**74.27**	72.19
4)	(1)+ Two-stage Prediction Aggregation	85.01	89.97	75.48	72.16	**72.72**
5)	All Non-local Features (1-4)	86.53	90.69	81.41	73.61	71.21

Table 3: The utility of non-local features. The system was trained on CoNLL03 data and tested on CoNNL03, MUC7 and Webpages. No single technique outperformed the rest on all domains. The combination of all techniques is the most robust.

5.2 Two-stage prediction aggregation

Context aggregation as done above can lead to excessive number of features. (Krishnan and Manning, 2006) used the intuition that some instances of a token appear in easily-identifiable contexts. Therefore they apply a baseline NER system, and use the resulting predictions as features in a second level of inference. We call the technique *two-stage prediction aggregation*. We implemented the token-majority and the entity-majority features discussed in (Krishnan and Manning, 2006); however, instead of document and corpus majority tags, we used relative frequency of the tags in a 1000 token window.

5.3 Extended prediction history

Both context aggregation and two-stage prediction aggregation treat all tokens in the text similarly. However, we observed that the named entities in the beginning of the documents tended to be more easily identifiable and matched gazetteers more often. This is due to the fact that when a named entity is introduced for the first time in text, a canonical name is used, while in the following discussion abbreviated mentions, pronouns, and other references are used. To break the symmetry, when using beamsearch or greedy left-to-right decoding, we use the fact that when we are making a prediction for token instance x_i, we have already made predictions $y_1, \ldots, y_{i-1}$ for token instances $x_1, \ldots, x_{i-1}$. When making the prediction for token instance x_i, we record the label assignment distribution for all token instances for the same token type in the previous 1000 words. That is, if the token instance is *"Australia"*, and in the previous 1000 tokens, the token type *"Australia"* was twice assigned the label *L-ORG* and three times the label *U-LOC*, then the prediction history feature will be: $(L-ORG:\frac{2}{5}; U-LOC:\frac{3}{5})$.

5.4 Utility of non-local features

Table 3 summarizes the results. Surprisingly, no single technique outperformed the others on all datasets. The extended prediction history method was the best on CoNLL03 data and MUC7 test set. Context aggregation was the best method for MUC7 development set and two-stage prediction was the best for Webpages. Non-local features proved less effective for MUC7 test set and the Webpages. Since the named entities in Webpages have less context, this result is expected for the Webpages. However, we are unsure why MUC7 test set benefits from non-local features much less than MUC7 development set. Our key conclusion is that no single approach is better than the rest and that the approaches are complimentary- their combination is the most stable and best performing.

6 External Knowledge

As we have illustrated in the introduction, NER is a knowledge-intensive task. In this section, we discuss two important knowledge resources– gazetteers and unlabeled text.

6.1 Unlabeled Text

Recent successful semi-supervised systems (Ando and Zhang, 2005; Suzuki and Isozaki, 2008) have illustrated that unlabeled text can be used to improve the performance of NER systems. In this work, we analyze a simple technique of using word clusters generated from unlabeled text, which has been shown to improve performance of dependency parsing (Koo et al., 2008), Chinese word segmentation (Liang, 2005) and NER (Miller et al., 2004). The technique is based on word class models, pioneered by (Brown et al., 1992), which hierarchically

	Component	CoNLL03 Test data	CoNLL03 Dev data	MUC7 Dev	MUC7 Test	Web pages
1)	Baseline	83.65	89.25	74.72	71.28	71.41
2)	(1) + Gazetteer Match	87.22	91.61	85.83	80.43	74.46
3)	(1) + Word Class Model	86.82	90.85	80.25	79.88	72.26
4)	All External Knowledge	88.55	92.49	84.50	83.23	74.44

Table 4: Utility of external knowledge. The system was trained on CoNLL03 data and tested on CoNNL03, MUC7 and Webpages.

clusters words, producing a binary tree as in Figure 2.

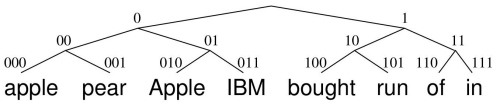


Figure 2: An extract from word cluster hierarchy.

The approach is related, but not identical, to distributional similarity (for details, see (Brown et al., 1992) and (Liang, 2005)). For example, since the words *Friday* and *Tuesday* appear in similar contexts, the Brown algorithm will assign them to the same cluster. Successful abstraction of both as a *day of the week*, addresses the data sparsity problem common in NLP tasks. In this work, we use the implementation and the clusters obtained in (Liang, 2005) from running the algorithm on the Reuters 1996 dataset, a superset of the CoNLL03 NER dataset. Within the binary tree produced by the algorithm, each word can be uniquely identified by its path from the root, and this path can be compactly represented with a bit string. Paths of different depths along the path from the root to the word provide different levels of word abstraction. For example, paths at depth 4 closely correspond to POS tags. Since word class models use large amounts of unlabeled data, they are essentially a semi-supervised technique, which we use to considerably improve the performance of our system.

In this work, we used path prefixes of length 4,6,10, and 20. When Brown clusters are used as features in the following sections, it implies that all features in the system which contain a word form will be duplicated and a new set of features containing the paths of varying length will be introduced. For example, if the system contains the feature *concatenation of the current token and the system prediction on the previous word*, four new features will be introduced which are concatenations of the previous prediction and the 4,6,10,20 length path-representations of the current word.

6.2 Gazetteers

An important question at the inception of the NER task was whether machine learning techniques are necessary at all, and whether simple dictionary lookup would be sufficient for good performance. Indeed, the baseline for the CoNLL03 shared task was essentially a dictionary lookup of the entities which appeared in the training data, and it achieves 71.91 F_1 score on the test set (Tjong and De Meulder, 2003). It turns out that while problems of coverage and ambiguity prevent straightforward lookup, injection of gazetteer matches as features in machine-learning based approaches is critical for good performance (Cohen, 2004; Kazama and Torisawa, 2007a; Toral and Munoz, 2006; Florian et al., 2003). Given these findings, several approaches have been proposed to automatically extract comprehensive gazetteers from the web and from large collections of unlabeled text (Etzioni et al., 2005; Riloff and Jones, 1999) with limited impact on NER. Recently, (Toral and Munoz, 2006; Kazama and Torisawa, 2007a) have successfully constructed high quality and high coverage gazetteers from Wikipedia.

In this work, we use a collection of 14 high-precision, low-recall lists extracted from the web that cover common names, countries, monetary units, temporal expressions, etc. While these gazetteers have excellent accuracy, they do not provide sufficient coverage. To further improve the coverage, we have extracted 16 gazetteers from Wikipedia, which collectively contain over 1.5M entities. Overall, we have 30 gazetteers (available for download with the system), and matches against

	Component	CoNLL03 Test data	CoNLL03 Dev data	MUC7 Dev	MUC7 Test	Web pages
1)	Baseline	83.65	89.25	74.72	71.28	71.41
2)	(1) + External Knowledge	88.55	92.49	84.50	83.23	74.44
3)	(1) + Non-local	86.53	90.69	81.41	73.61	71.21
4)	**All Features**	**90.57**	**93.50**	**89.19**	**86.15**	**74.53**
5)	All Features (train with dev)	90.80	N/A	89.19	86.15	74.33

Table 5: End system performance by component. Results confirm that NER is a knowledge-intensive task.

each one are weighted as a separate feature in the system (this allows us to trust each gazetteer to a different degree). We also note that we have developed a technique for injecting non-exact string matching to gazetteers, which has marginally improved the performance, but is not covered in the paper due to space limitations. In the rest of this section, we discuss the construction of gazetteers from Wikipedia.

Wikipedia is an open, collaborative encyclopedia with several attractive properties. (1) It is kept updated manually by it collaborators, hence new entities are constantly added to it. (2) Wikipedia contains redirection pages, mapping several variations of spelling of the same name to one canonical entry. For example, *Suker* is redirected to an entry about *Davor Šuker*, the Croatian footballer (3) The entries in Wikipedia are manually tagged with categories. For example, the entry about the *Microsoft* in Wikipedia has the following categories: *Companies listed on NASDAQ; Cloud computing vendors; etc.*

Both (Toral and Munoz, 2006) and (Kazama and Torisawa, 2007a) used the free-text description of the Wikipedia entity to reason about the entity type. We use a simpler method to extract high coverage and high quality gazetteers from Wikipedia. By inspection of the CoNLL03 shared task annotation guidelines and of the training set, we manually aggregated several categories into a higher-level concept (not necessarily NER type). When a Wikipedia entry was tagged by one of the categories in the table, it was added to the corresponding gazetteer.

6.3 Utility of External Knowledge

Table 4 summarizes the results of the techniques for injecting external knowledge. It is important to note that, although the world class model was learned on the superset of CoNLL03 data, and although the Wikipedia gazetteers were constructed based on CoNLL03 annotation guidelines, these features proved extremely good on all datasets. Word class models discussed in Section 6.1 are computed offline, are available online[1], and provide an alternative to traditional semi-supervised learning. It is important to note that the word class models and the gazetteers and independednt and accumulative. Furthermore, despite the number and the gigantic size of the extracted gazetteers, the gazeteers alone are not sufficient for adequate performance. When we modified the CoNLL03 baseline to include gazetteer matches, the performance went up from 71.91 to 82.3 on the CoNLL03 test set, below our baseline system's result of 83.65. When we have injected the gazetteers into our system, the performance went up to 87.22. Word class model and nonlocal features further improve the performance to 90.57 (see Table 5), by more than 3 F_1 points.

Dataset	Stanford-NER	LBJ-NER
MUC7 Test	80.62	85.71
MUC7 Dev	84.67	87.99
Webpages	72.50	74.89
Reuters2003 test	87.04	90.74
Reuters2003 dev	92.36	93.94

Table 6: Comparison: token-based F_1 score of LBJ-NER and Stanford NER tagger across several domains

7 Final System Performance Analysis

As a final experiment, we have trained our system both on the training and on the development set, which gave us our best F_1 score of 90.8 on the CoNLL03 data, yet it failed to improve the performance on other datasets. Table 5 summarizes the performance of the system.

Next, we have compared the performance of our

[1] http://people.csail.mit.edu/maestro/papers/bllip-clusters.gz

system to that of the Stanford NER tagger, across the datasets discussed above. We have chosen to compare against the Stanford tagger because to the best of our knowledge, it is the best publicly available system which is trained on the same data. We have downloaded the Stanford NER tagger and used the strongest provided model trained on the CoNLL03 data with distributional similarity features. The results we obtained on the CoNLL03 test set were consistent with what was reported in (Finkel et al., 2005). Our goal was to compare the performance of the taggers across several datasets. For the most realistic comparison, we have presented each system with a raw text, and relied on the system's sentence splitter and tokenizer. When evaluating the systems, we matched against the gold tokenization ignoring punctuation marks. Table 6 summarizes the results. Note that due to differences in sentence splitting, tokenization and evaluation, these results are not identical to those reported in Table 5. Also note that in this experiment we have used token-level accuracy on the CoNLL dataset as well. Finally, to complete the comparison to other systems, in Table 7 we summarize the best results reported for the CoNLL03 dataset in literature.

	System	Resources Used	F_1
+	LBJ-NER	Wikipedia, Nonlocal Features, Word-class Model	90.80
-	(Suzuki and Isozaki, 2008)	Semi-supervised on 1G-word unlabeled data	89.92
-	(Ando and Zhang, 2005)	Semi-supervised on 27M-word unlabeled data	89.31
-	(Kazama and Torisawa, 2007a)	Wikipedia	88.02
-	(Krishnan and Manning, 2006)	Non-local Features	87.24
-	(Kazama and Torisawa, 2007b)	Non-local Features	87.17
+	(Finkel et al., 2005)	Non-local Features	86.86

Table 7: Results for CoNLL03 data reported in the literature. publicly available systems marked by +.

8 Conclusions

We have presented a simple model for NER that uses expressive features to achieve new state of the art performance on the Named Entity recognition task. We explored four fundamental design decisions: text chunks representation, inference algorithm, using non-local features and external knowledge. We showed that BILOU encoding scheme significantly outperforms BIO and that, surprisingly, a conditional model that does not take into account interactions at the output level performs comparably to beamsearch or Viterbi, while being considerably more efficient computationally. We analyzed several approaches for modeling non-local dependencies and found that none of them clearly outperforms the others across several datasets. Our experiments corroborate recently published results indicating that word class models learned on unlabeled text can be an alternative to the traditional semi-supervised learning paradigm. NER proves to be a knowledge-intensive task, and it was reassuring to observe that knowledge-driven techniques adapt well across several domains. We observed consistent performance gains across several domains, most interestingly in Webpages, where the named entities had less context and were different in nature from the named entities in the training set. Our system significantly outperforms the current state of the art and is available to download under a research license.

Apendix– wikipedia gazetters & categories

1)**People**: *people, births, deaths*. Extracts 494,699 Wikipedia titles and 382,336 redirect links. 2)**Organizations**: *cooperatives, federations, teams, clubs, departments, organizations, organisations, banks, legislatures, record labels, constructors, manufacturers, ministries, ministers, military units, military formations, universities, radio stations, newspapers, broadcasters, political parties, television networks, companies, businesses, agencies*. Extracts 124,403 titles and 130,588 redirects. 3)**Locations**: *airports, districts, regions, countries, areas, lakes, seas, oceans, towns, villages, parks, bays, bases, cities, landmarks, rivers, valleys, deserts, locations, places, neighborhoods*. Extracts 211,872 titles and 194,049 redirects. 4)**Named Objects**: *aircraft, spacecraft, tanks, rifles, weapons, ships, firearms, automobiles, computers, boats*. Extracts 28,739 titles and 31,389 redirects. 5)**Art Work**: *novels, books, paintings, operas, plays*. Extracts 39,800 titles and 34037 redirects. 6)**Films**: *films, telenovelas, shows, musicals*. Extracts 50,454 titles and 49,252 redirects. 7)**Songs**: *songs, singles, albums*. Extracts 109,645 titles and 67,473 redirects. 8)**Events**: *playoffs, championships, races, competitions, battles*. Extracts 20,176 titles and 15,182 redirects.

References

R. K. Ando and T. Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *ACL*.

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

X. Carreras, L. Màrquez, and L. Padró. 2003. Learning a perceptron-based named entity chunker via online recognition feedback. In *CoNLL*.

H. Chieu and H. T. Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of CoNLL*.

W. W. Cohen. 2004. Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In *KDD*.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

L. Edward. 2007. Finding good sequential model structures using output transformations. In *EMNLP)*.

O. Etzioni, M. J. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

J. R. Finkel, T. Grenager, and C. D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*.

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. 2003. Named entity recognition through classifier combination. In *CoNLL*.

Y. Freund and R. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

J. Kazama and K. Torisawa. 2007a. Exploiting wikipedia as external knowledge for named entity recognition. In *EMNLP*.

J. Kazama and K. Torisawa. 2007b. A new perceptron algorithm for sequence labeling with non-local features. In *EMNLP-CoNLL*.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *ACL*.

V. Krishnan and C. D. Manning. 2006. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *ACL*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*. Morgan Kaufmann.

P. Liang. 2005. Semi-supervised learning for natural language. *Masters thesis, Massachusetts Institute of Technology*.

S. Miller, J. Guinness, and A. Zamanian. 2004. Name tagging with word clusters and discriminative training. In *HLT-NAACL*.

A. Molina and F. Pla. 2002. Shallow parsing using specialized hmms. *The Journal of Machine Learning Research*, 2:595–613.

A. Niculescu-Mizil and R. Caruana. 2005. Predicting good probabilities with supervised learning. In *ICML*.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *NIPS*.

L. R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *IEEE*.

E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI*.

N. Rizzolo and D. Roth. 2007. Modeling discriminative global inference. In *ICSC*.

D. Roth and D. Zelenko. 1998. Part of speech tagging using a network of linear separators. In *COLING-ACL*.

H. Shen and A. Sarkar. 2005. Voting between multiple data representations for text chunking. *Advances in Artificial Intelligence*, pages 389–400.

J. Suzuki and H. Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *ACL*.

E. Tjong, K. and F. De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

A. Toral and R. Munoz. 2006. A proposal to automatically build and maintain gazetteers for named entity recognition by using wikipedia. In *EACL*.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*.

J. Veenstra. 1999. Representing text chunks. In *EACL*.

T. Zhang and D. Johnson. 2003. A robust risk minimization based named entity recognition system. In *CoNLL*.

Automatic Selection of High Quality Parses Created By a Fully Unsupervised Parser

Roi Reichart
ICNC
The Hebrew University
roiri@cs.huji.ac.il

Ari Rappoport
Institute of computer science
The Hebrew University
arir@cs.huji.ac.il

Abstract

The average results obtained by unsupervised statistical parsers have greatly improved in the last few years, but on many specific sentences they are of rather low quality. The output of such parsers is becoming valuable for various applications, and it is radically less expensive to create than manually annotated training data. Hence, automatic selection of high quality parses created by unsupervised parsers is an important problem.

In this paper we present PUPA, a *POS-based Unsupervised Parse Assessment* algorithm. The algorithm assesses the quality of a parse tree using POS sequence statistics collected from a batch of parsed sentences. We evaluate the algorithm by using an unsupervised POS tagger and an unsupervised parser, selecting high quality parsed sentences from English (WSJ) and German (NEGRA) corpora. We show that PUPA outperforms the leading previous parse assessment algorithm for supervised parsers, as well as a strong unsupervised baseline. Consequently, PUPA allows obtaining high quality parses without any human involvement.

1 Introduction

In unsupervised parsing an algorithm should uncover the syntactic structure of an input sentence without using any manually created structural training data. The last decade has seen significant progress in this field of research (Klein and Manning, 2002; Klein and Manning, 2004; Bod, 2006a; Bod, 2006b; Smith and Eisner, 2006; Seginer, 2007).

Many NLP systems use the output of supervised parsers (e.g., (Kwok et al., 2001) for QA, (Moldovan et al., 2003) for IE, (Punyakanok et al., 2008) for SRL, (Srikumar et al., 2008) for Textual Inference and (Avramidis and Koehn, 2008) for MT). To achieve good performance, these parsers should be trained on large amounts of manually created training data from a domain similar to that of the sentences they parse (Lease and Charniak, 2005; McClosky and Charniak, 2008). In the highly variable Web, where many of these systems are used, it is very difficult to create a representative corpus for manual annotation. The high cost of manual annotation of training data for supervised parsers imposes a significant burden on their usage.

A possible answer to this problem can be provided by high quality parses produced by unsupervised parsers that require little to no manual efforts for their training. These parses can be used either as input for applications, or as training material for modern supervised parsers whose output will in turn be used by applications.

Although unsupervised parser results improve, the quality of many of the parses they produce is still too low for such goals. For example, the Seginer (2007) parser achieves an F-score of 75.9% on the WSJ10 corpus and 59% on the NEGRA10 corpus, but the percentage of individual sentences with an F-score of 100% is 21.5% for WSJ10 and 11% for NEGRA10. When requirements are relaxed, only asking for an F-score higher than 85%, percentage is still low, 42% for WSJ10 and 15% for NEGRA10.

In this paper we address the task of a fully unsupervised assessment of high quality parses cre-

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 156–164,
Boulder, Colorado, June 2009.

ated by an unsupervised parser. The assessment should be unsupervised in order to avoid the problems mentioned above with manually trained supervised parsers. Assessing the quality of a learning algorithm's output and selecting high quality instances has been addressed for supervised algorithms (Caruana and Niculescu-Mizil, 2006) and specifically for supervised parsers (Yates et al., 2006; Reichart and Rappoport, 2007; Kawahara and Uchimoto, 2008; Ravi et al., 2008). Moreover, it has been shown to be valuable for supervised parser adaptation between domains (Sagae and Tsujii, 2007; Kawahara and Uchimoto, 2008; Chen et al., 2008). However, as far as we know the present paper is the first to address the task of unsupervised assessment of the quality of parses created by *unsupervised* parsers.

Our *POS-based Unsupervised Parse Assessment* (PUPA) algorithm uses statistics about POS tag sequences in a batch of parsed sentences[1]. The constituents in the batch are represented using the POS sequences of their yield and of the yields of neighboring constituents. Constituents whose representation is frequent in the output of the parser are considered to be of a high quality. A score for each range of constituent length is calculated, reflecting the robustness of statistics used for the creation of the constituents of that length. The final sentence score is a weighted average of the scores calculated for each constituent length. The score thus integrates the quality of short and long constituents into one score reflecting the quality of the whole parse tree.

PUPA provides a quality score for every sentence in a parsed sentences set. An NLP application can then decide if to use a parse or not, according to its own definition of a high quality parse. For example, it can select every sentence whose score is above some threshold, or the k top scored sentences. The selection strategy is application dependent and is beyond the scope of this paper.

The unsupervised parser we use is the Seginer (2007) incremental parser[2], which achieves state-of-the-art results without using manually created POS tags. The POS tags we use are induced by the unsupervised tagger of (Clark, 2003)[3]. Since both tagger and parser do not require any manual annotation, PUPA identifies high quality parses without any human involvement.

The incremental parser of (Seginer, 2007) does not give any prediction of its output quality, and extracting such a prediction from its internal data structures is not straightforward. Such a prediction can be given by supervised parsers in terms of the parse likelihood, but this was shown to be of medium quality (Reichart and Rappoport, 2007). While the algorithms of Yates et al. (2006), Kawahara and Uchimoto (2008) and Ravi et al. (2008) are supervised (Section 3), the ensemble based SEPA algorithm (Reichart and Rappoport, 2007) can be applied to unsupervised parsers in a way that preserves the unsupervised nature of the selection task.

To compare between two algorithms, we use each of them to assess the quality of the sentences in English and German corpora (WSJ and NEGRA)[4]. We show that for every sentence length (up to 20) the quality of the top scored k sentences according to PUPA is higher than the quality of SEPA's list (for every k). As in (Reichart and Rappoport, 2007), the quality of a set selected from the parser's output is evaluated using two measures: constituent F-score[5] and average sentence F-score.

Section 2 describes the PUPA algorithm, Section 3 discusses previous work, and Sections 4 and 5 present the evaluation setup and results.

2 The POS-based Unsupervised Parse Assessment (PUPA) Algorithm

In this section we detail our parse assessment algorithm. Its input consists of a set I of parsed sentences, which in our evaluation scenario are produced by an unsupervised parser. The algorithm assigns each parsed sentence a score reflecting its quality.

[1] The algorithm can be used with supervised POS taggers and parsers, but we focus here on the fully unsupervised scenario, which is novel and more useful. For completeness of analysis, we experimented with PUPA using a supervised POS tagger (see Section 5). Using PUPA with supervised parsers is left for future work.

[2] www.seggu.net/ccl.

[3] www.cs.rhul.ac.uk/home/alexc/RHUL/Downloads.html, the neyessenmorph model.

[4] This is in contrast to algorithms for selection from the results of supervised constituency parsers, which were evaluated only for English (Yates et al., 2006; Reichart and Rappoport, 2007; Ravi et al., 2008).

[5] This is the traditional parsing F-score.

The algorithm has three steps. First, the words in I are POS tagged (in our case, using the fully unsupervised POS induction algorithm of Clark (2003)). Second, POS statistics about the constituents in I are collected. Finally, a quality score is calculated for each parsed sentence in I using the POS statistics. In the following we detail the last two steps.

Collecting POS statistics. In its second step, the algorithm collects statistics about the constituents in the input set I. Recall that the *yield* of a constituent is the set of words covered by it. The PUPA *constituent representation (PCR)* consists of three features: (1) the ordered POS tag sequence of the constituent's yield, (2) the constituents' right context, and (3) the constituents' left context.

We define *context* to be the *leftmost and rightmost* POS tags in the yield of the *neighbor* of the constituent (if there is only one POS tag in the neighbor's yield, this POS tag is the context). For the right and left contexts we consider the right and left neighbors respectively. A constituent C_1 is the right neighbor of a constituent C_2 if C_1 is the highest level constituent such that the first word in the yield of C_1 comes immediately after the last word in the yield of C_2. A constituent C_1 is the left neighbor of a constituent C_2 if C_1 is the highest level constituent such that the first word in the yield of C_2 comes immediately after the last word in the yield of C_1.

Figure 1 shows an example, an unlabeled tree for the sentence 'I will give you the ball'. The tree has 6 constituents (C0-C5). C3 and C4 have both right and left neighbors. For C3, the POS sequence of its yield is POS2, POS3 , the left neighbor is C1 and thus the left context is POS1, and the right neighbor is C4 and thus the right context is POS4. Note that the left and right neighbors of C3 have only one POS tag in their yield and therefore this POS tag is the context. For C4 the yield is POS4, the left neighbor is C3 (and thus the left context is POS2,POS3), and the right neighbor is C5 (and thus the right context is POS5,POS6). C1, whose yield is POS1, has only a right neighbor, C2, and thus its right context is POS2,POS6 and its left context is NULL. C2 and C5 (whose yields are POS2, POS3, POS4, POS5, POS6 for C2 and POS5, POS6 for C5) have only a left neighbor. For C2, this is C1 (and the context is POS1) while for C5 this is C4 (with the context POS4).

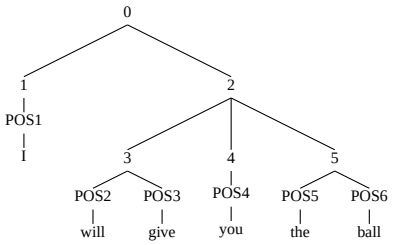


Figure 1: An example parse tree for contexts and neighbors (see text).

The right context of both constituents is NULL. As all sentence level constituents, C0 has no neighbors, and thus both its left and right contexts are NULL.

We have also explored other representations of left and right contexts based on the POS tags of their yields. In these, we represented the left/right neighbor using only the leftmost/rightmost POS tags of its yield or other subsets of the yield's POS tags. These variations produced lower quality results than the main variant above in our experiments, which were for English and German. Exploring the suitability of our representation for other languages is left for future research.

Score computation. The third and last step of the algorithm is a second pass over I for computing a quality score for each parse tree.

Short constituents tend to be more frequent than long ones. In order not to distort our score due to parsing errors in short constituents, PUPA computes the grade using a division into lengths, in three steps. First, constituents are assigned to bins according to their length, each bin containing the constituents of a certain range of lengths. Denote this range by W (for width), and the number of bins by $N(W)$. For example, in our experiments the longest possible constituent is of length 20, so we can take $W = 5$, resulting in $N(W) = 4$: bin 1 for constituents of length 1-5, bin 2 for constituents of length 6-10, and so on for bins 3, 4.

The score of bin_i is given by

$$(1)\ BinScore(bin_i) = \sum_{t=2}^{t=X}(X - t + 2) \cdot \frac{|C_t^i|}{|C^i|}$$

Where X is the maximal number of occurrences of constituents in the bin that we consider as important for the score (see below for its selection), $|C_t^i|$ is the number of constituents in bin i occurring at

least t times in the batch of parsed sentences, and $|C^i|$ is the number of constituents in bin i. In words, the score is a weighted average: the fraction of the constituents in the bin occuring at least 2 times (with weight X), plus the fraction of the constituents in the bin occuring at least 3 times (with weight $X-1$), etc, until the fraction of the constituents in the bin occuring at least X times (with weight 2).

A score for the division into N bins is given by

(2) $Score(N(W)) = \frac{\sum_{i=1}^{N(W)} BinScore(bin_i)}{Z \cdot M}$

Where Z is the maximum bin score (according to (1)) and M is the number of bins containing at least one constituent. If, for example, $N(W) = 4$ and there is no constituent whose length is between 11 and 15 then bin number 3 is empty. If every other bin contains at least one constituent, $M = 3$.

To get a final score for the parse tree of sentence S that is independent of a specific bin division, we sum the scores of the various bin division:

(3) $PupaScore(S) = \frac{\sum_{W=1}^{W=Y} Score(N(W))}{Y}$

where Y is the length of S (which is also its maximum bin width). $PupaScore$ thus takes values in the $[0, 1]$ range.

In equation (1), if, for example, $X = 20$ then the weight of the fraction of the bin's constituents occurring at least 2 times is 20 while the weight of the fraction of the constituents occurring at least 10 times is 12 and of the fraction of constituents occurring at least 20 times is 2. We consider the number of times a constituent appears in a batch to be an indication of its correctness. The difference between 3 and 2 occurrences is therefore more indicative than the difference between 20 and 19 occurrences. More generally, the more times a constituent occurs, the less indicative any additional appearance is.

In equation (2) we give all bins the same weight. Short constituents are more frequent and are generally more likely to be correct. However, the correctness of long constituents is an indication that the parser has a correct interpretation of the tree structure and that it is likely to create a high quality tree. The usage of equal bin weights was done to balance the tendency of parse trees to have more short constituents.

Parameters. PUPA has two parameters: X, the maximal number of occurrences considered in equation (1), and P, the number of POS tags induced by the unsupervised POS tagger. In the following we present the unsupervised technique we used to tune these parameters.

Figure 2 shows $nc(t)$, the number of constituents appearing at least t times in WSJ20 (left) and NEGRA20 (right). For both corpora, the pattern is shown when using 5 POS tags ($P = 5$, solid line) and 50 POS tags (P = 50, dashed line). The distribution obeys Zipf's law: many constituents appear a small number of times while a few constituents appear a large number of times. We denote the t value where the slope changes from steep to moderate by t_{elbow}. Practically, we approximate the 'real' elbow value and define t_{elbow} to be the smallest t for which $nc(t+1) - nc(t) = 1$. When $P = 5$, t_{elbow} is 32 for WSJ and 19 for NEGRA. When P = 50, t_{elbow} is 15 for WSJ and 9 for NEGRA.

The number of constituents appearing more than t_{elbow} times is considerably smaller than the number of constituents appearing t_{elbow} times or less. Therefore, the fact that a constituent appears $t_{elbow} + S$ times (for a positive integer S) is not a better indication of its quality than the fact that it appears t_{elbow} times. We thus select X to be t_{elbow}.

The graphs also demonstrate that for both corpora, t_{elbow} for $P = 50$ is smaller than t_{elbow} for $P = 5$. Generally, t_{elbow} is a monotonically decreasing function of P. Lower t_{elbow} values imply that PUPA would be less distinctive between constituents quality (see equation (1); recall that $X = t_{elbow}$). We thus want to select the P value that maximizes t_{elbow}. We therefore minimize P. t_{elbow} values for $P \in \{3, \ldots, 10\}$ are very similar. Indeed, PUPA achieves its best performance for $P \in \{3, \ldots, 10\}$ and it is insensitive to the selection of P in this range. In Section 5 we report results with $P = 5$.

3 Related Work

Unsupervised parsing has been explored for several decades (see (Klein, 2005) for a recent review). Recently, unsupervised parsing algorithms have for the first time outperformed the right branching heuristic baseline for English. These include CCM (Klein and Manning, 2002), the DMV and DMV+CCM models (Klein and Manning, 2004), (U)DOP based mod-

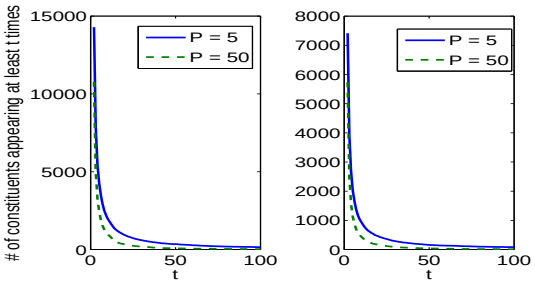


Figure 2: Number of constituents appearing at least t times ($nc(t)$) as a function of t. Shown are WSJ (left) and NEGRA (right), where constituents are represented according to PUPA's PCR with 5 POS tags ($P = 5$, solid line) or 50 POS tags ($P = 50$, dashed line).

els (Bod, 2006a; Bod, 2006b), an exemplar based approach (Dennis, 2005), guiding EM using contrastive estimation (Smith and Eisner, 2006), and the incremental parser of Seginer (2007) that we use in this work. To obtain good results, manually created POS tags are used as input in all of these algorithms except Seginer's, which uses plain text.

Quality assessment of a learning algorithm's output and selection of high quality instances have been addressed for supervised algorithms (see (Caruana and Niculescu-Mizil, 2006) for a survey) and specifically for supervised constituency parsers (Yates et al., 2006; Reichart and Rappoport, 2007; Ravi et al., 2008). For dependency parsing in a corpus adaptation scenario, (Kawahara and Uchimoto, 2008) built a binary classifier that classifies each parse in the parser's output as reliable or not. To do that, they selected 2500 sentences from the parser's output, compared them to their manually created gold standard, and used accurate (inaccurate) parses as positive (negative) examples for the classifier. Their approach is supervised and the features used by the classifier are dependency motivated .

As far as we know, the present paper is the first to address the task of selecting high quality parses from the output of unsupervised parsers. The algorithms of Yates et al. (2006), Kawahara and Uchimoto (2008) and Ravi et al. (2008) are supervised, performing semantic analysis of the parse tree and gold standard-based calssification, respectively. However, the SEPA algorithm of Reichart and Rappoport (2007), an algorithm for supervised constituency parsers, can be applied to unsupervised parsers in a way that preserves the unsupervised nature of the selection task. In Section 5 we provide a detailed comparison between PUPA and SEPA showing the first to be superior. Below is a brief description of the SEPA algorithm.

The input of the SEPA algorithm consists of a parsing algorithm A, a training set, and a test set (which in the unsupervised case might be the same set). The algorithm provides, for each of the test set's parses generated by A when trained on the full training set, a grade assessing the parse quality, on a continuous scale between 0 to 100. The quality grade is calculated in the following way: N random samples of size S are sampled from the training data and used for training the parsing algorithm A. In that way N committee members are created. Then, each of the test sentences is parsed by each of the N committee members and an agreement score ranging from 0 to 100 between the committee members is calculated. All unsupervised parsers mentioned above (including the Seginer parser), have a training phase where parameter values are estimated from unlabeled data. SEPA can thus be applied to the unsupervised case.

Automatic selection of high quality parses has been shown to improve parser adaptation. Sagae and Tsujii (2007) and Kawahara and Uchimoto (2008) applied a self-training protocol to a parser adaptation scenario but used only high quality parses to retrain the parser. In the first work, high quality parses were selected using an ensemble method, while in the second a binary classifier was used (see above). The first system achieved the highest score in the CoNLL 2007 shared task on domain adaptation of dependency parsers, and the second system improved over the basic self-training protocol. Chen et al. (2008) parsed target domain sentences and used short dependencies information, which is often accurate, to adapt a dependency parser to the Chinese language.

Automatic quality assessment has been extensively explored for machine translation (Ueffing and Ney, 2007) and speech recognition (Koo et al., 2001). Other NLP tasks where it has been explored include semi-supervised relation extraction (Rosenfeld and Feldman, 2007), IE (Culotta and McCallum, 2004), QA (Chu-Carroll et al., 2003), and dialog systems (Lin and Weng, 2008).

The idea of representing a constituent by its yield

and (a different definition of) context is used by the CCM unsupervised parsing model (Klein and Manning, 2002). As far as we know the current work is the first to use unsupervised POS tags for the selection of high quality parses.

4 Evaluation Setup

We experiment with sentences of up to 20 words from the English WSJ Penn Treebank (WSJ20, 25236 sentences, 225126 constituents) and the German NEGRA corpus (Brants, 1997) (NEGRA20, 15610 sentences, 108540 constiteunts), both containing newspaper texts.

The unsupervised parsers of the kind addressed in this paper output unlabeled parse trees. To evaluate the quality of a single parse tree with respect to another, we use the unlabeled F-score ($UF = \frac{2 \cdot UR \cdot UP}{UR+UP}$), where UR and UP are unlabeled recall and unlabeled precision respectively.

Following the unsupervised parsing literature, multiple brackets and brackets covering a single word are not counted, but the sentence level bracket is. We exclude punctuation and null elements according to the scheme of (Klein, 2005).

The performance of unsupervised parsers markedly degrades as sentence length increases. For example, the Average sentence F–score for WSJ sentences of length 10 is 71.4% compared to 58.5 for sentences of length 20 (the numbers for NEGRA are 48.2% and 36.9%). We therefore evaluate PUPA (and the baseline) for sentences of a given length. We do this for every sentence of length 2-20 in WSJ20 and NEGRA20.

For every sentence length L, we use PUPA and the baseline algorithm (SEPA) to give a quality score to each of the sentences of that length in the experimental corpus. We then compare the quality of the top k parsed sentences according to each algorithm. We do this for every k from 1 to the number of sentences of length L.

Following Reichart and Rappoport (2007), we use two measures to evaluate the quality of a *set* of parses: the *constituent F-score* (the traditional F-score used in the parsing literature), and the *average F-score* of the parses in the set. In the first measure we treat the whole set as a bag of constituents. Each constituent is marked as correct (if it appears in the gold standard parses of the set) or erroneous (if it does not). Then, recall, precision and F-score are calculated over these constituents. In the second measure, the constituent F-score of each of the parses in the set is computed, and then results are averaged.

There are applications that use individual constituents from the output of a parser while others need the whole parse tree. For example, if the selected set is used for training supervised parsers such as the Collins parser (Collins, 1999), which collects constituent statistics, the constituent F-score of the selected set is the important measure. In applications such as the syntax based machine translation model of (Yamada and Knight, 2001), a low quality tree might lead to errorenous translation of the sentence. For such applications the average F-score is more indicative. These measures thus represent complementary aspects of a set quality and we consider both of them.

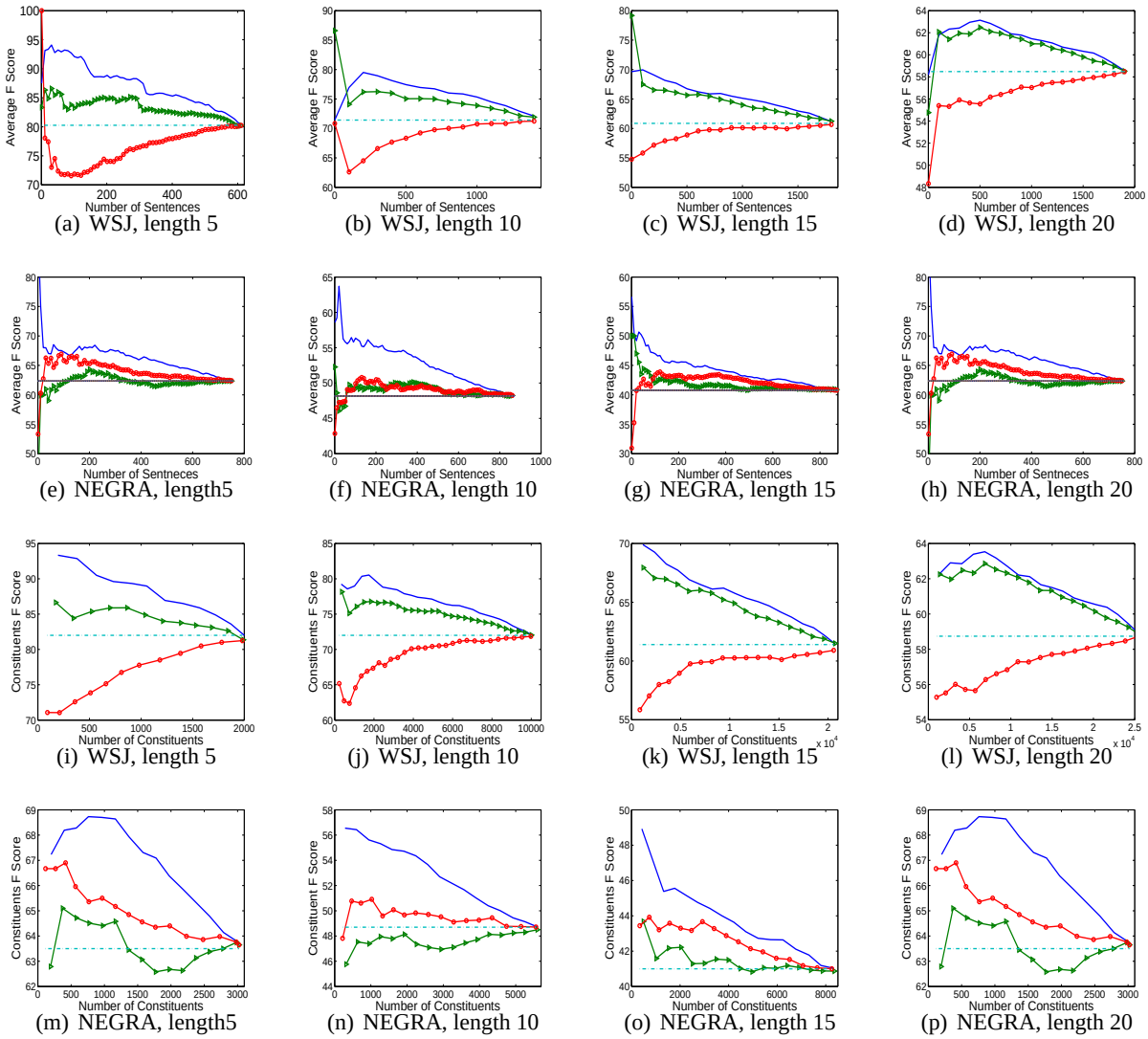
The parser we use is the incremental parser of (Seginer, 2007), POS tags are induced using the unsupervised POS tagger of ((Clark, 2003), neyessenmorph model). In each experiment, the tagger was trained with the raw sentences of the experiment corpus, and then the corpus words were POS tagged.

The output of the unsupervised POS tagger depends on a random initialization. We ran the tagger 5 times, each time with a different random initialization, and then ran PUPA with its output. The results we report for PUPA are the average over these 5 runs. Random selection results (given for reference) were also averages over 5 samples.

PUPA 's parameter estimation is completely unsupervised (see Section 2). No development data was used to tune its parameters.

A 200 sentences development set from each corpus was used for calibrating the parameters of the SEPA algorithm. Based on the analysis of SEPA performance with different assignments of its parameters given by Reichart and Rappoport (2007) (see Section 3), we ran the SEPA algorithm with sample size (SEPA parameter S) of 30% and 80%, and with 2 – 10 committee members (N)[6]. The optimal parameters were $N = 10$,$S = 80$ for WSJ20, and

[6]We tried higher N values but observed no improvements in SEPA's performance.



(a) WSJ, length 5 (b) WSJ, length 10 (c) WSJ, length 15 (d) WSJ, length 20
(e) NEGRA, length5 (f) NEGRA, length 10 (g) NEGRA, length 15 (h) NEGRA, length 20
(i) WSJ, length 5 (j) WSJ, length 10 (k) WSJ, length 15 (l) WSJ, length 20
(m) NEGRA, length5 (n) NEGRA, length 10 (o) NEGRA, length 15 (p) NEGRA, length 20

Figure 3: **In all graphs:** PUPA: solid line. SEPA: line with triangles. MC: line with circles. Random selection is presented for reference as a dotted line. **Top two rows:** Average F-score for PUPA, SEPA and MC for sentences from WSJ (top row) and NEGRA (bottom row). **Bottom two rows:** Constituents F-score for PUPA, SEPA and MC for sentences from WSJ (top row) and NEGRA (bottom row). Results are presented for sentence lengths of 5,10,15 and 20 (patterns for other sentence lengths between 2 and 20 are very similar). PUPA is superior in all cases. The graphs for PUPA and SEPA show a downward trend because parsed sentences were sorted according to score, which correlates positively with F-score (unlike MC). The graphs converge because on the extreme right all test sentences were selected.

$N = 10$, $S = 30$ for NEGRA20.

We also compare PUPA to a baseline selecting the sentences with the lowest number of constituents. Since the number of constituents is an indication of the complexity of the syntactic structure of a sentence, it is reasonable to assume that selecting the sentences with the lowest number of constituents is a good selection strategy. We denote this baseline by MC (for minimum constituents).

The incremental parser does not give any prediction of its output quality as supervised generative parsers do. We are thus not able to compare to such a score.

5 Results

Figure 3 shows Average F-score and Constituents F-score results for PUPA SEPA and MC, for sentences

of lengths 5,10,15 and 20 in WSJ20 and NEGRA20. The top two rows are for Average F-score (top row: WSJ, bottom row: NEGRA), while the bottom two rows are for Constituents F-score (top row: WSJ, bottom row: NEGRA).

PUPA and SEPA are both better than random selection for both corpora for every sentence length. The MC baseline is better than random selection only for NEGRA (in which case it outperforms SEPA). For WSJ, however, random selection is a better strategy than MC.

It is clear from the graphs that PUPA outperforms SEPA and MC in all experimental conditions. We observed very similar patterns in all other sentence lengths in WSJ20 and NEGRA20 for both Average F-score and Constituent F-score. In other words, for every sentence length in both corpora, PUPA outperforms SEPA and MC in terms of both measures. we present our results per sentence length to deprive the possibility that PUPA is useful only for short sentences or that it prefers sentences whose syntactic structure is not complex (i.e. with a small number of constituents, like MC).

Table 1 shows that the same pattern of results holds when evaluating on the whole corpus (WSJ20 or NEGRA20) without any sentence length restriction.

Note that while PUPA is a fully unsupervised algorithm, SEPA requires a few hundreds of sentences for its parameters tuning.

The main result of this paper is for sentences whose length is up to 20 words (note that most unsupervised parser literature reports numbers for sentences up to length 10). We have also ran the experiments for the remaining length range, 20-40. For NEGRA, PUPA is superior over MC up to length 36, and both are much better than SEPA. For WSJ, PUPA and SEPA both outperform MC, but SEPA is a bit better than PUPA. When evaluating on the whole corpus (i.e. without sentence length restriction, like in Table 1) PUPA is superior over both SEPA and MC for WSJ40 and NEGRA40.

For completeness of analysis we also experimented in the condition where PUPA uses gold standard POS tags as input. The number of these tags is 35 for WSJ and 57 for NEGRA. Interestingly, PUPA achieves in this condition the same performance as when using the same number of POS tags induced by an unsupervised POS tagger. Since PUPA's performance for a smaller number of POS tags is better (see our parameter tuning discussion above), the bottom line is that PUPA pefers using induced POS tags over gold POS tags.

	5%	10%	20%	30%	40%	50%
WSJ20						
PUPA	**82.75**	**79.34**	**75.77**	**73.46**	**71.68**	**70.3**
SEPA	78.68	75.7	72.64	70.72	69.54	68.58
MC	76.75	74.6	72.1	70.35	68.97	67.77
NEGRA20						
PUPA	**70.66**	**67.06**	**61.89**	**58.75**	**56.6**	**54.73**
SEPA	66.19	62.75	59.41	57.16	55.23	53.7
MC	69.41	65.79	60.87	58.08	55.9	54.36

Table 1: Average F–score for the top k% of constituents selected from WSJ20 (up) and NEGRA20 (down). No sentence length restriction is imposed. Results presented for PUPA , SEPA and MC. Average F–score of random selection is 66.55 (WSJ20) and 47.05 (NEGRA20). PUPA is superior over all methods.

6 Conclusions

We introduced PUPA, an algorithm for unsupervised parse assessment that utilizes POS sequence statistics. PUPA is a fully unsupervised algorithm whose parameters can be tuned in an unsupervised manner. Experimenting with the Seginer unsupervised parser and Clark's unsupervised POS tagger on English and German corpora, PUPA was shown to outperform both the leading parse assessment algorithm for supervised parsers (SEPA, even when its parameters are tuned on manually annotated development data) and a strong baseline (MC).

Using PUPA, we extracted high quality parses from the output of a parser which requires raw text as input, using POS tags induced by an unsupervised tagger. PUPA thus provides a way of obtaining high quality parses without any human involvement.

For future work, we intend to use parses selected by PUPA from the output of unsupervised parsers as training data for supervised parsers, and in NLP applications that use parse trees. A challenge for the first direction is the fact that state of the art supervised parsers require labeled parse trees, while modern unsupervised parsers create unlabeled trees. Combining PUPA with algorithms for labeled parse trees induction (Haghighi and Klein, 2006; Reichart and Rappoport, 2008) is a one direction to overcome this challenge. We also intend to use PUPA to assess the quality of parses created by supervised parsers.

References

Eleftherios Avramidis and Philipp Koehn, 2008. Enriching Morphologically Poor Languages for Statistical Machine Translation. *ACL '08*.

Rens Bod, 2006a. An All-Subtrees Approach to Unsupervised Parsing. *ACL '06*.

Rens Bod, 2006b. Unsupervised Parsing with U-DOP. *CoNLL X*.

Thorsten Brants, 1997. The NEGRA Export Format. *CLAUS Report, Saarland University*.

Rich Caruana and Alexandru Niculescu-Mizil, 2006. An Empirical Comparison of Supervised Learning Algorithms. *ICML '06*.

Jennifer Chu-Carroll, Krzysztof Czuba, John Prager and Abraham Ittycheriah, 2003. In Question Answering, Two Heads Are Better Than One. *HLT-NAACL '03*.

Wenliang Chen, Youzheng Wu and Hitoshi Isahara, 2008. Learning Reliable Information for Dependency Parsing Adaptation. *Coling '08*.

Alexander Clark, 2003. Combining Distributional and Morphological Information for Part of Speech Induction. *EACL '03*.

Michael Collins, 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Aron Culotta and Andrew McCallum, 2004. Confidence Estimation for Information Extraction. *HLT-NAACL '04*.

Simon Dennis, 2005. An Exemplar-based Approach to Unsupervised Parsing. *Proceedings of the 27th Conference of the Cognitive Science Society*.

Aria Haghighi and Dan Klein, 2006. Prototype-driven Grammar Induction. *ACL '06*.

Daisuke Kawahara and Kiyotaka Uchimoto 2008. Learning Reliability of Parses for Domain Adaptation of Dependency Parsing. *IJCNLP '08*.

Dan Klein and Christopher Manning, 2002. A Generative Constituent-Context Model for Improved Grammar Induction. *ACL '02*.

Dan Klein and Christopher Manning, 2004. Corpus-based Induction of Syntactic Structure: Models of Dependency and Constituency. *ACL '04*.

Dan Klein, 2005. *The Unsupervised Learning of Natural Language Structure.* Ph.D. thesis, Stanford University.

Myoung–Wan Koo, Chin-Hui Lee and Biing–Hwang Juang 2001. Speech Recognition and Utterance Verification Based on a Generalized Confidence Score. *IEEE Transactions on Speech and Audio Processing*, 9(8):821–832.

Cody Kwok, Oren Etzioni and Daniel S. Weld, 2001. Scaling Question Answering to the Web. *WWW '01*.

Matthew Lease and Eugene Charniak, 2005. Towards a Syntactic Account of Punctuation. *IJCNLP '05*.

Feng Lin and Fuliang Weng, 2008. Computing Confidence Scores for All Sub Parse Trees. *ACL '08, short paper*.

David McClosky and Eugene Charniak, 2008. Self-Training for Biomedical Parsing. *ACL '08, short paper*.

Dan Moldovan, Christine Clark, Sanda Harabagiu and Steve Maiorano, 2003. Cogex: A Logic Prover for Question Answering. *HLT-NAACL '03*.

Vasin Punyakanok and Dan Roth and Wen-tau Yih, 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Computational Linguistics*, 34(2):257-287.

Sujith Ravi, Kevin Knight and Radu Soricut, 2008. Automatic Prediction of Parser Accuracy. *EMNLP '08*.

Roi Reichart and Ari Rappoport, 2007. An Ensemble Method for Selection of High Quality Parses. *ACL '07*.

Roi Reichart and Ari Rappoport, 2008. Unsupervised Induction of Labeled Parse Trees by Clustering with Syntactic Features. *COLING '08*.

Benjamin Rosenfeld and Ronen Feldman, 2007. Using Corpus Statistics on Entities to Improve Semi–Supervised Relation Extraction From The WEB. *ACL '07*.

Kenji Sagae and Junichi Tsujii, 2007. Dependency Parsing and Domain Adaptation with LR Models and Parser Ensemble. *EMNLP-CoNLL '07*.

Yoav Seginer, 2007. Fast Unsupervised Incremental Parsing. *ACL '07*.

Vivek Srikumar, Roi Reichart, Mark Sammons, Ari Rappoport and Dan Roth, 2008. Extraction of Entailed Semantic Relations Through Syntax-based Comma Resolution. *ACL '08*.

Noah A. Smith and Jason Eisner, 2006. Annealing Structural Bias in Multilingual Weighted Grammar Induction. *ACL '06*.

Nicola Ueffing and Hermann Ney, 2007. Word-Level Confidence Estimation for Machine Translation. *Computational Linguistics*, 33(1):9–40.

Kenji Yamada and Kevin Knight, 2001. A Syntax-Based Statistical Translation Model. *ACL '01*.

Alexander Yates, Stefan Schoenmackers and Oren Etzioni, 2006. Detecting Parser Errors Using Web-based Semantic Filters . *EMNLP '06*.

The NVI Clustering Evaluation Measure

Roi Reichart
ICNC
Hebrew University of Jerusalem
roiri@cs.huji.ac.il

Ari Rappoport
Institute of Computer Science
Hebrew University of Jerusalem
arir@cs.huji.ac.il

Abstract

Clustering is crucial for many NLP tasks and applications. However, evaluating the results of a clustering algorithm is hard. In this paper we focus on the evaluation setting in which a gold standard solution is available. We discuss two existing information theory based measures, V and VI, and show that they are both hard to use when comparing the performance of different algorithms and different datasets. The V measure favors solutions having a large number of clusters, while the range of scores given by VI depends on the size of the dataset. We present a new measure, NVI, which normalizes VI to address the latter problem. We demonstrate the superiority of NVI in a large experiment involving an important NLP application, grammar induction, using real corpus data in English, German and Chinese.

1 Introduction

Clustering is a major technique in machine learning and its application areas. It lies at the heart of unsupervised learning, which has great potential advantages over supervised learning. This is especially true for NLP, due to the high efforts and costs incurred by the human annotations required for training supervised algorithms. Recent NLP problems addressed by clustering include POS induction (Clark, 2003; Goldwater and Griffiths, 2007), word sense disambiguation (Shin and Choi, 2004), semantic role labeling (Baldewein et al., 2004), pitch accent type disambiguation (Levow, 2006) and grammar induction (Klein, 2005).

Evaluation of clustering results is a challenging task. In this paper we address the *external measures* setting, where a correct assignment of elements to *classes* is available and is used for evaluating the quality of another assignment of the elements into *clusters*. Many NLP works have used external clustering evaluation measures (see Section 2).

Recently, two measures have been proposed that avoid many of the weaknesses of previous measures and exhibit several attractive properties (see Sections 2 and 3): the VI measure (Meila, 2007) and the V measure (Rosenberg and Hirschberg, 2007). However, each of these has a serious drawback. The possible values of VI lie in [0, $2log\ N$], where N is the size of the clustered dataset. Hence it has limited use when comparing performance on different datasets. V measure values lie in $[0, 1]$ regardless of the dataset, but the measure strongly favors a clustering having many small clusters. In addition, V does not have many of the attractive properties of VI.

This paper has two contributions. First, we propose the NVI measure, a normalization of VI which guarantees that the score of clusterings that VI considers good lies in [0,1], regardless of dataset size. Most of VI's attractive properties are retained by NVI.

Second, we compare the behavior of V, VI and NVI in various situations to the desired behavior and to each other. In particular, we show that V gives high scores to clusterings with a large number of clusters even when they are of low quality. We demonstrate this both in a synthetic example (Section 5) and in the evaluation (in three languages) of a difficult NLP problem, labeled parse tree induc-

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 165–173,
Boulder, Colorado, June 2009.

tion (Section 6). We show that in both cases, NVI constitutes a better clustering evaluation measure.

2 Previous Evaluation Measures

A large number of clustering quality measures have been proposed. Here we briefly survey the three main types, mapping based measures, counting pairs measures and information theory based measures.

We first review some terminology (Meila, 2007; Rosenberg and Hirschberg, 2007). In a **homogeneous** clustering, every cluster contains only elements from a single class. In a **complete** clustering, all elements of each class are assigned to the same cluster. The **perfect** solution is the fully homogeneous and complete clustering. We will illustrate the behavior of some measures using three extreme cases: the **single cluster** case, in which all data elements are put in the same single cluster; the **singletons** case, in which each data element is put in a cluster of its own; and the **no knowledge** case, in which the class distribution within each cluster is identical to the class distribution in the entire dataset. If the single cluster solution is not the perfect one, the no knowledge solution is the worst possible solution. Throughout the paper, the number of data elements to be clustered is denoted by **N**.

Mapping based measures are based on a post-processing step in which each cluster is mapped to a class. Among these are: L (Larsen, 1999), D (Van Dongen, 2000), misclassification index (MI) (Zeng et al., 2002), H (Meila, 2001), clustering F-measure (Fung et al., 2003) and micro-averaged precision and recall (Dhillon et al., 2003). As noted in (Rosenberg and Hirschberg, 2007), these measures evaluate not only the quality of the proposed clustering but also of the mapping scheme. Different mapping schemes can lead to different quality scores for the same clustering. Moreover, even when the mapping scheme is fixed, it can lead to not evaluating the entire membership of a cluster and not evaluating every cluster (Meila, 2007).

Counting pairs measures are based on a combinatorial approach which examines the number of pairs of data elements that are clustered similarly in the reference and proposed clustering. Among these are Rand Index (Rand, 1971), Adjusted Rand Index (Hubert and Arabie, 1985), Γ statistic (Hubert and Schultz, 1976), Jaccard (Milligan et al., 1983), Fowlkes-Mallows (Fowlkes and Mallows, 1983) and Mirkin (Mirkin, 1996).

Meila (2007) described a number of problems with such measures. The most acute one is that their values are unbounded, making it hard to interpret their results. The problem can be solved by transformations adjusting their values to lie in $[0, 1]$, but the adjusted measures suffer from severe distributional problems, again limiting their usability in practice.

Information-theoretic (IT) based measures are those addressed in this work. The measures in this family suffer neither from the problems associated with mappings, since they evaluate the entire membership of each cluster and not just a mapped portion, nor from the distributional problems of the counting pairs measures.

Zhao and Karypis (2001) define *Purity* and *Entropy* as follows:
$Purity = \sum_{r=1}^{k} \frac{1}{N} max_i(n_r^i)$
$Entropy = \sum_{r=1}^{k} \frac{n_r}{N}(-\frac{1}{logq}\sum_{i=1}^{q}\frac{n_r^i}{n_r}log(\frac{n_r^i}{n_r}))$
where q is the number of classes, k the number of clusters, n_r cluster r's size, and n_r^i is the number of elements in class i assigned to cluster r.

Both measures are good measures for homogeneity (Purity increases and Entropy decreases when homogeneity increases). However, they do not evaluate completeness at all. The singletons solution is thus considered optimal even if in fact it is of very low quality.

Dom (2001) proposed the Q measure, the sum of a homogeneity term $H(C|K)$ and a model cost term calculated using a coding theory argument:
$Q(C, K) = H(C|K) + \frac{1}{N}\sum_{k=1}^{|k|} log\binom{h(k)+|C|-1}{|C|-1}$
where C are the correct classes, K are the induced clusters and $h(k)$ is the number of elements in cluster k. Dom also presented a normalized version of the Q measure (called Q_2) whose range is $(0, 1]$ and gives higher scores to clusterings that are preferable. As noted by (Rosenberg and Hirschberg, 2007), the Q measure does not explicitly address the completeness of the suggested clustering. Due to the cost term, if two clusterings have the same $H(C|K)$ value, the model prefers the one with the lower number of clusters, but the trade-off between homogeneity and completeness is not explicitly addressed.

In the next section we describe the V and VI mea-

sures, which are IT measures that explicitly assess both the homogeneity and completeness of the clustering solution.

BCubed (Bagga and Baldwin, 1998) is an attractive measure that addresses both completeness and homogeneity. It does not explicitly use IT concepts and avoids mapping. In this paper we focus on V and VI; a detailed comparison with BCubed is out of our scope here and will be done in future work.

Several recent NLP papers used clustering techniques and evaluation measures. Examples include (Finkel and Manning, 2008), using VI, Rand index and clustering F-score for evaluating coreference resolution; (Headden et al., 2008), using VI, V, greedy 1-to-1 and many-to-1 mapping for evaluating unsupervised POS induction; (Walker and Ringger, 2008), using clustering F-score, the adjusted Rand index, V, VI and Q_2 for document clustering; and (Reichart and Rappoport, 2008), using greedy 1-to-1 and many-to-1 mappings for evaluating labeled parse tree induction.

Schulte im Walde (2003) used clustering to induce semantic verb classes and extensively discussed non-IT based clustering evaluation measures. Pfitzner et al. (2008) presented a comparison of clustering evaluation measures (IT based and others). While their analysis is extensive, their experiments were confined to artificial data. In this work, we experiment with a complex NLP application using large real datasets.

3 The V and VI Measures

The V (Rosenberg and Hirschberg, 2007) and VI (Meila, 2007) measures are IT based measures. In this section we give a detailed description of these measures and analyze their properties.

Notations. The partition of the N data elements into classes is denoted by $C = \{c_1, \ldots, c_{|C|}\}$. The clustering solution is denoted by $K = \{k_1, \ldots, k_{|K|}\}$. $A = \{a_{ij}\}$ is a $|C| \times |K|$ contingency matrix such that a_{ij} is the number of data elements that are members of class c_i and are assigned by the algorithm to cluster k_j.

As other IT measures, V and VI assume that the elements in the dataset are taken from a known distribution (both assume the uniform distribution), and thus the classes and clusters can be treated as random variables. When assuming the uniform distribution, the probability of an event (a class or a cluster) is its relative size, so $p(c) = \sum_{k=1}^{|K|} \frac{a_{ck}}{N}$ and $p(k) = \sum_{c=1}^{|C|} \frac{a_{ck}}{N}$. Under this assumption we can talk about the entropies $H(C)$ and $H(K)$ and the conditional entropies $H(C|K)$ and $H(K|C)$:

$$H(C) = -\sum_{c=1}^{|C|} \frac{\sum_{k=1}^{|K|} a_{ck}}{N} log \frac{\sum_{k=1}^{|K|} a_{ck}}{N}$$

$$H(K) = -\sum_{k=1}^{|K|} \frac{\sum_{c=1}^{|C|} a_{ck}}{N} log \frac{\sum_{c=1}^{|C|} a_{ck}}{N}$$

$$H(C|K) = -\sum_{k=1}^{|K|} \sum_{c=1}^{|C|} \frac{a_{ck}}{N} log \frac{a_{ck}}{\sum_{c=1}^{|C|} a_{ck}}$$

$$H(K|C) = -\sum_{k=1}^{|K|} \sum_{c=1}^{|C|} \frac{a_{ck}}{N} log \frac{a_{ck}}{\sum_{k=1}^{|K|} a_{ck}}$$

In Section 2 we defined the concepts of homogeneity and completeness. In order to satisfy the homogeneity criterion, each cluster must be contained in a certain class. This results in the minimization of the conditional entropy of the classes given the clusters, $H(C|K) = 0$. In the least homogeneous solution, the conditional entropy is maximized, and $H(C|K) = H(C)$. Similarly, in order to satisfy the completeness criterion, each class must be contained in a certain cluster, which results in the minimization of the conditional entropy of the clusters given the classes, $H(K|C) = 0$. In the least complete solution, the conditional entropy is maximized, and $H(K|C) = H(K)$.

The VI measure. Variation of information (VI) is defined as follows:

$$VI(C, K) = H(C|K) + H(K|C).$$

In the least homogeneous (complete) clustering, the values of $H(C|K)$ ($H(K|C)$) are maximal. As a clustering solution becomes more homogeneous (complete), the values of $H(C|K)$ ($H(K|C)$) decrease to zero. Consequently, *lower* VI values imply better clustering solutions. In the perfect solution, both $H(C|K) = 0$ and $H(K|C) = 0$ and thus $VI = 0$. For the least homogeneous and complete clustering solution, where knowing the cluster tells nothing about the class and vise versa, $VI = H(C) + H(K)$.

As a result, the range of values that VI takes is dataset dependent, and the numbers themselves tell

us nothing about the quality of the clustering solution (apart from a score of 0, which is given to the best possible solution).

A bound for VI values is a function of the maximum number of clusters in C or K, denoted by k^*. This is obtained when each cluster contains a single element, and $k^* = N$. Thus, $VI \in [0, 2logN]$. Consequently, the range of VI values is dataset dependent and unbounded when datasets change. This means that it is hard to use VI to compare the performance of a clustering algorithm across datasets.

An apparent simple solution to this problem would be to normalize VI by $2logk^*$ or $2logN$, so that its values would lie in $[0, 1]$. We discuss this at the end of the next section.

VI has two useful properties. First, it satisfies the **metric axioms**, that is: $VI(C, K) \geq 0$, $VI(C, K) = VI(K, C)$, $VI(C_1, C_2) + VI(C_2, C_3) \geq VI(C_1, C_3)$. This gives an intuitive understanding of the relation between VI values.

Second, it is **convexly additive**. This means that if K is obtained from C by splitting C_j into clusters $K_j^1, \ldots, K_j^m$, $\hat{H}(K_j) = -\sum_{i=1}^{m} P(K_j^i|C_j) logP(K_j^i|C_j)$, then $VI(C, K) = P(C_j)\hat{H}(K_j)$. This property guarantees that all changes to VI are local; the impact of splitting or merging clusters is limited only to those clusters involved, and its size is relative to the size of these clusters.

The V measure. The V measure uses homogeneity (h) and completeness (c) terms as follows:

$$h = \begin{cases} 1 & H(C) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & H(C) \neq 0 \end{cases}$$

$$c = \begin{cases} 1 & H(K) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & H(K) \neq 0 \end{cases}$$

$$V = \frac{2hc}{h + c}$$

In the least homogeneous clustering, $H(C|K)$ is maximal, at $H(C|K) = H(C)$. In this case h reaches its minimum value, which is 0. As homogeneity increases $H(C|K)$ values decrease. For the most homogeneous clustering, $H(C|K) = 0$ and $h = 1$. The same considerations hold for c, which ranges between 0 (for the least complete clustering) and 1 (for a complete clustering). Since V is defined to be the harmonic mean of h and c, V values lie in $[0, 1]$. Consequently, it can be used to compare the performance of clustering algorithms across datasets. Higher V values imply better clusterings.

Unlike VI, V does not satisfy the metric axioms and is not convexly additive. The range of values it can get does not depend on dataset size.

Extreme cases for the two measures. In the single cluster solution $H(C|K) = H(C)$ and $H(K|C) = 0$, and thus $V = 0$ (the worst possible score) and $VI = H(C)$. If there is indeed only a single class, then $VI = 0$, the best possible score, which is the correct behavior. VI behaves better than V here.

The singletons solution is a fully homogeneous clustering in which $H(C|K) = 0$. The score of each measure depends on the completeness of the solution. The completeness of a singletons clustering increases with the number of classes. In the extreme case where every element is assigned to a unique class ($|C| = |K| = N$) singletons is also complete, $H(K|C) = 0$, and $V(C, K) = 1, VI(C, K) = 0$. Both measures exhibit the correct behavior.

If there are classes that contain many elements, singletons is far from being complete and should be treated as a low quality solution. Again, in the singletons solution $VI = H(K|C)$. Suppose that the number of clusters is fixed. When the number of classes increases, this value decreases, which is what we want. When the number of classes decreases, the score increases, which is again the correct behavior. In Section 5 we show that this desired behavior shown by VI is not shown by V.

Both measures treat the no knowledge solution as the worst one possible: $V = 0$, and $VI = H(C) + H(K)$.

4 Normalized Variation of Information

In this section we define NVI, a normalization of VI. NVI is N-independent and its values for clusterings considered as good by VI lie in $[0, 1]$. Hence, NVI can be used to compare clustering performance across datasets. We show that NVI keeps the convex additivity property of VI but not its metric axioms.

Definition. We define NVI to be:

$$NVI(C,K) = \begin{cases} \frac{H(C|K)+H(K|C)}{H(C)} & H(C) \neq 0 \\ H(K) & H(C) = 0 \end{cases}$$

We define NVI to be $H(K)$ when $H(C) = 0$ to satisfy the requirements that NVI values decrease as C and K become more similar and that NVI would be 0 when they are identical[1].

Range and extreme cases. Like VI, NVI decreases as the clustering becomes more complete and more homogeneous. For the perfect solution, $NVI = 0$. In both the single cluster and the no knowledge solutions, $H(C|K) = H(C)$. Thus, in the former case $NVI = 1$, and in the latter $NVI = 1 + \frac{H(K)}{HC} \geq 1$.

For the singletons clustering case, $NVI = \frac{H(K|C)}{H(C)}$. Suppose that the number of clusters is fixed. When the number of classes increases, the numerator decreases and the denominator increases, and hence the score decreases. In other words, as the real solution gets closer to the singletons solution, the score decreases, which is the correct behavior. When the number of classes decreases, the score increases, which is again the correct behavior.

For any pair of clusterings K_1 and K_2, $VI(C,K_1) > VI(C,K_2)$ iff $NVI(C,K_1) > VI(C,K_2)$. This implies that only clustering solutions whose VI scores are better (i.e., numerically lower) than the score of the single cluster solution will be scored lower than 1 by NVI.

Note that NVI is meant to be used when there is a 'correct' reference solution. In this case $H(C)$ is constant, so the property above holds. In this sense, VI is more general, allowing us to compare any three clustering solutions even when we do not have a correct reference one.

To summarize:

1. All clusterings considered by VI to be of high quality (i.e., better than the single cluster solution) are scored by NVI in the range of $[0, 1]$.

2. All clusterings considered by VI to be of lower quality than the single cluster solution are scored higher than 1 by NVI.

3. The ordering of scores between solutions given by VI is preserved by NVI.

4. The behavior of NVI on the extreme cases is the desired one.

Useful properties. In Section 3 we saw that VI has two useful properties, satisfying the metric axioms and being convexly additive. NVI is not symmetric since the term in its denominator is $H(C)$, the entropy of the correct class assignment. Thus, it does not satisfy the metric axioms. Being convexly additive, however, is preserved. In the class splitting scenario (see convex additivity definition in Section 3) it holds that $NVI(C,K) = \frac{P(C_j)\hat{H}(K_j)}{H(C)}$. That is, like for VI, the impact of splitting or merging a cluster on NVI is limited only to those clusters involved, and its size is relative to the size of these clusters. Meila (2007) derived various interesting properties of VI from the convex additivity property. These properties generally hold for NVI as well.

$H(K)$ **normalization.** Normalizing by $H(C)$ takes into consideration the complexity of the correct clustering. Another normalization option would be to normalize by $H(K)$, which represents the induced clustering complexity. This normalization does not guarantee that the scores of the 'good' clusterings lie in a data-independent range.

Let us define NVIK(C,K) to be $\frac{VI(C,K)}{H(K)}$ if $H(K) > 0$ and $H(C)$ if $H(K) = 0$. Recall that in order for NVIK to be 0 iff C and K are identical, we must require that $NVIK = H(C)$ when $H(K) = 0$. In the no knowledge case, $NVIK = \frac{H(C)+H(K)}{H(K)} = \frac{H(C)}{H(K)} + 1 > 1$. In the single cluster solution, however, $NVIK = H(C)$ (since in this case $H(K) = 0$) which ranges in $[0, logN]$. This is a serious drawback of NVIK. In Section 6 we empirically show an additional drawback of NVIK.

$logN$ **normalization.** Another possible normalization of VI is by $2logN$ (or $2logk^*$), which is an upper bound on VI values. However, this results in the values of the measure being dependent on dataset size, so results on datasets with different sizes again cannot be compared. For example, take any C and K and split each element into two. All entropy values, and the quality of the solution, are preserved, but the scores given to the two K's (before and after

[1] $H(C) = 0$ iff C consists of a single class, and therefore $H(C) = H(K) = 0$ iff C (K) consists of a single class (cluster).

7	1	1	1	0	0	0	0	0	0
0	7	1	1	1	0	0	0	0	0
0	0	7	1	1	1	0	0	0	0
0	0	0	7	1	1	1	0	0	0
0	0	0	0	7	1	1	1	0	0
0	0	0	0	0	7	1	1	1	0
0	0	0	0	0	0	7	1	1	1
1	0	0	0	0	0	0	7	1	1
1	1	0	0	0	0	0	0	7	1
1	1	1	0	0	0	0	0	0	7

	V	VI	NVI	NVIK
Singletons	0.667	2.303	**1**	0.5
Solution R	0.587	1.88	**0.81**	0.81

Table 1: The clustering matrix of solution R (top), and the scores given to it and to the singletons solution by the four measures (bottom). Although solution R is superior, the score given by V to the singletons solution is much higher. NVI exhibits the most preferable behavior (recall that higher V values are better, as opposed to the other three measures).

the split) by such a normalized VI would be different. Since $H(C)$ is preserved, the scores given by NVI to the two K's are identical.

5 Problematic V Behavior Example

In this section we provide a synthetic example that demonstrates an undesireable behavior of V (and NVIK) not manifested by VI and NVI. Specifically, V favors solutions with a large number of clusters, giving them higher scores than to solutions that are evidently superior. In addition, the score given to the singletons solution is high in absolute terms.

To present the example, we use the matrix representation A of a clustering solution defined in Section 3. The entries in row i sum to the number of elements in class i, while those in column j sum to the number of elements in cluster j.

Suppose that we have 100 elements assigned to 10 classes such that there are 10 elements in each class. We consider two clustering solutions: the singletons solution, and solution R whose matrix is shown in Table 1 (top). Like the real solution, solution R also has 10 clusters each having 10 elements. Solution R is not very far from the correct solution, since each cluster has 7 elements of the same class, and the three other elements in a cluster are taken from a different class each and can be viewed as 'noise'. Solution R is thus much better than the singletons solution. In order not to rely on our own opinion, we have performed a simple human judgment experiment with 30 subjects (university graduates in different fields), all of whom preferred solution R[2].

The scores given by V, VI, NVI and NVIK to the two solutions are shown in Table 1 (bottom). V scores solution R as being worse than the singletons solution, and gives the latter a number that's relatively high in absolute terms (0.667). VI exhibits qualitatively correct behavior, but the numbers it uses are hard to interpret since they are N-dependent. NVI scores solution R as being better than singletons, and its score is less than 1, indicating that it might be a good solution.

6 Grammar Induction Experiment

In this section we analyse the behavior of V, VI, NVI and NVIK using a highly non-trivial NLP application with large real datasets, the unsupervised labeled parse tree induction (LTI) algorithm of (Reichart and Rappoport, 2008). We focus on the labeling that the algorithm finds for parsing constituents, which is a clustering of constituents.

Summary of result. We show that V gives about the same score to a labeling that uses thousands of labels and to labelings in which the number of labels (dozens) is identical or smaller than the number of labels in the reference evaluation set (an annotated corpus). Contrary to V, both NVI and VI give much better scores to the solutions having a smaller number of labels.

It could be argued that the total number of 'real' labels in the data is indeed large (e.g., because every verb exhibits its own syntactic patterns) and that a small number of labels is just an arbitrary decision of the corpus annotators. However, most linguistic theories agree that there is a prototypical level of generalization that uses concepts such as Noun Phrase and Verb Phrase, a level which consists of at most dozens of labels and is strongly manifested by real language data. Under these accepted assumptions, the scoring behavior of V is unreasonable.

[2]We must rely on people's expectations, since the whole point in this area is that clustering quality cannot be formalized in an objective, application-independent way.

	MDL+SC (T labels)					MDL+SC (P labels)					MDL labels				
Corpus	L	$=1$	<10	$<10^2$	$\geq 10^2$	L	$=1$	<10	$<10^2$	$\geq 10^2$	L	$=1$	<10	$<10^2$	$\geq 10^2$
WSJ10	26	0	0	3	23	8	0	0	0	8	2916	2282	2774	2864	52
NEGRA10	22	0	2	12	10	6	0	0	1	5	1202	902	1114	1191	11
CTB10	24	1	4	11	13	9	1	2	4	5	1050	816	993	1044	6

Table 2: The number of elements (constituents) covered by the clusters (labels) produced by the MDL+SC (T or P labels) and MDL clusterings. L is the total number of labels. Shown are the number of clusters having one element, less than 10 elements, less than 100 elements, and more than 100 elements. It is evident that MDL induces a sparse clustering with many clusters that annotate very few constituents.

	V			VI			NVI			NVIK		
Corpus	MDL	T	P	MDL	T	P	MDL	T	P	MDL	T	P
WSJ10	0.4	0.44	0.41	3.83	2.32	1.9	2.21	1.34	**1.1**	0.81	0.86	1.2
NEGRA10	0.47	0.5	0.5	2.56	1.8	1.4	1.51	1.1	**0.83**	0.76	0.96	1.1
CTB10	0.42	0.42	0.45	3	2.22	1.85	1.72	1.26	**1.1**	0.87	1.1	1.25

Table 3: V, VI, NVI and NVIK values for MDL and MDL+SC with T or P labels. V gives the three clusterings very similar scores. NVIK prefers MDL labeling. NVI and VI both show the expected qualitative behavior, favoring MDL+SC clustering with P labels. The most preferable scores are those of NVI, whose numbers are also the easiest to interpret.

The experiment. The LTI algorithm has three stages: bracketing, initial labeling, and label clustering. Bracketing is done from raw text using the unsupervised incremental parser of (Seginer, 2007). Initial labeling is done using the BMM model (Borensztajn and Zuidema, 2007), which aims at minimizing the grammar description length (MDL). Finally, labels are clustered to a desired number of labels using the k-means algorithm with syntactic features extracted from the initially labeled trees. We refer to this stage as MDL+SC (for 'syntactic clustering'). Using a mapping-based evaluation with two different mapping functions, the LTI algorithm was shown to outperform previous work on unsupervised labeled parse tree induction.

The MDL clustering step induces several thousand labels for corpora of several tens of thousands of constituents. The role of the SC step is to generalize these labels using syntactic features. There are two versions of the SC step. In one, the number of clusters is identical to the number of labels in the gold standard annotation of the experimental corpus. This set of labels is called T (for target) labels. In the other SC version, the number of labels is the minimum number of labels required to annotate more than 95% of the constituents in the gold standard annotation of the corpus. This set of labels is called P (for prominent) labels. Since constituent labels follow the Zipfian distribution, P is much smaller than T.

In this paper we run the LTI algorithm and evaluate its labeling quality using V, VI, NVI and NVIK. We compare the quality of the clustering induced by the first clustering step alone (the MDL clustering) to the quality of the clustering induced by the full algorithm (i.e., first applying MDL and then clustering its output using the SC algorithm for T or P labels)[3].

We follow the experimental setup in (Reichart and Rappoport, 2008), running the algorithm on English, German and Chinese corpora: the WSJ Penn Treebank (English), the Negra corpus (Brants, 1997) (German), and version 5.0 of the Chinese Penn Treebank (Xue et al., 2002). In each corpus, we used the sentences of length at most 10,[4] numbering 7422 (WSJ10), 7542 (NEGRA10) and 4626 (CTB10).

The characteristics of the induced clusterings are shown in Table 2[5]. The table demonstrates the fact that MDL labeling, while perhaps capturing the

[3]Note that our evaluation here has nothing to do with the evaluation done in (Reichart and Rappoport, 2008), which provided a comparison of the full grammar induction results between different algorithms, using mapping-based measures. We evaluate the labeling stages alone.

[4]Excluding punctuation and null elements, according to the scheme of (Klein, 2005).

[5]The number of MDL labels in the table differs from their numbers, since we report the number of unique MDL labels used for annotating correct constituents in the parser's output, while they report the number of unique labels used for annotating *all* constituents in the parser's output.

salient level of generalization of the data in its leading clusters, is extremely noisy. For WSJ10, for example, 2282 of the 2916 unique labels annotate only one constituent, and 2774 labels label less than 10 constituents. These 2774 labels annotate 14.4% of compared constituents, and the 2864 labels that annotate less than 100 constituents each, cover 30.7% of the compared constituents (these percentages are not shown in the table). In other words, MDL is not a solution in which almost all of the mass is concentrated in the few leading clusters; its tail occupies a large percentage of its mass.

MDL patterns for NEGRA10 and CTB10 are very similar. For MDL+SC with T or P labels, most of the induced labels annotate 100 constituents or more. We thus expect MDL+SC to provide better clustering than MDL; a good clustering evaluation measure should reflect this expectation.

Table 3 shows V, VI, NVI and NVIK scores for MDL and MDL+SC (with T or P labels). For all three corpora, V values are almost identical for the MDL and the MDL+SC schemes. This is in contrast to VI and NVI values that strongly prefer the MDL+SC clusterings, fitting our expectations (recall that for these measures, the lower the score, the better the clustering). Moreover, VI and NVI prefer MDL+SC with P labels, which again accords with our expectations, since P labels were defined as those that are more salient in the data (see above).

The patterns of NVI and VI are identical, since $NVI = \frac{VI}{H(C)}$ and $H(C)$ is independent of the induced clustering. However, the numbers given by NVI are easier to interpret than those given by VI. The latter are basically meaningless, conveying nothing about clustering quality. The former are quite close to 1, telling us that clustering quality is not that good but not horrible either. This makes sense, because the overall quality of the labeling induction algorithm is indeed not that high: using one-to-one mapping (the more forgiving mapping), the accuracy of the labels induced by MDL+SC is only 45–72% (Reichart and Rappoport, 2008).

NVIK, the normalization of VI with $H(K)$, is worse even than V. This measure (which also gives lower scores to better clusterings) prefers the MDL over MDL+SC labels. This is a further justification of our decision to define NVI by normalizing VI by $H(C)$ rather than by $H(K)$.

Corpus	$H(C)$	$H(K)$		
		MDL	T	P
WSJ10	1.73	4.72	2.7	1.58
NEGRA10	1.69	3.36	1.87	1.29
CTB10	1.76	3.45	2.1	1.48

Table 4: Class ($H(C)$) and cluster ($H(K)$) entropy for MDL and MDL+SC with T or P labels. $H(C)$ is cluster independent. $H(K)$ increases with the number of clusters.

Table 4 shows the $H(C)$ and $H(K)$ values in the experiment. While $H(C)$ is independent of the induced clustering and is thus constant for a given annotated corpus, $H(K)$ monotonically increases with the number of induced clusters. Since both NVIK and the completeness term of V are normalized by $H(K)$, these measures prefer clusterings with a large number of clusters even when many of these clusters provide useless information.

7 Conclusion

Unsupervised clustering evaluation is important for various NLP tasks and applications. Recently, the importance of the completeness and homogeneity as evaluation criteria for such clusterings has been recognized. In this paper we addressed the two measures that address these criteria: VI (Meila, 2007) and V (Rosenberg and Hirschberg, 2007).

While VI has many useful properties, the range of values it can take is dataset dependent, which makes it unsuitable for comparing clusterings of different datasets. This imposes a serious restriction on the measure usage. We presented NVI, a normalized version of VI, which does not have this restriction and still retains some of its useful properties.

Using experiments with both synthetic data and a complex NLP application, we showed that the V measure prefers clusterings with many clusters even when these are clearly of low quality. VI and NVI do not exhibit such behavior, and the numbers given by NVI are easier to interpret than those given by VI.

In future work we intend to explore more of the properties of NVI and use it in other real NLP applications.

References

Amit Bagga and Breck Baldwin, 1998. Entity-based cross-document coreferencing using the vector space

model. *ACL 98*.

Ulrike Baldewein, Katrin Erk, Sebastian Pado, and Detlef Prescher 2004. Semantic role labeling with similarity based generalization using EM–based clustering. *Senseval '04*.

Thorsten Brants, 1997. The NEGRA export format. *CLAUS Report, Saarland University*.

Gideon Borensztajn and Willem Zuidema, 2007. Bayesian model merging for unsupervised constituent labeling and grammar induction. Technical Report, ILLC. http: //staff.science.uva.nl/∼gideon/

Alexander Clark, 2003. Combining distributional and morphological information for part of speech induction. *EACL '03*.

I. S. Dhillon, S. Mallela, and D. S. Modha, 2003. Information theoretic co-clustering. *KDD '03*.

Byron E. Dom, 2001. An information-theoretic external cluster validity measure . *Journal of American statistical Association*,78:553–569.

Jenny Rose Finkel and Christopher D. Manning, 2008. Enforcing transitivity in coreference resolution. *ACL '08*.

E.B Fowlkes and C.L. Mallows, 1983. A method for comparing two hierarchical clusterings. *Journal of American statistical Association*,78:553–569.

Benjamin C. M. Fung, Ke Wang, and Martin Ester, 2003. Hierarchical document clustering using frequent itemsets. *SIAM International Conference on Data Mining '03*.

Sharon Goldwater and Thomas L. Griffiths, 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. *ACL '07*.

William P. Headden, David McClosky, and Eugene Charniak, 2008. Evaluating unsupervised part-of-speech tagging for grammar induction. *COLING '08*.

L. Hubert and P. Arabie, 1985. Comparing partitions. *Journal of Classification*, 2:193–218.

L. Hubert and J. Schultz, 1976. Quadratic assignment as a general data analysis strategy. *British Journal of Mathematical and Statistical Psychology*, 29:190–241.

Dan Klein, 2005. The unsupervised learning of natural language structure. Ph.D. thesis, Stanford University.

Bjornar Larsen and Chinatsu Aone, 1999. Fast and effective text mining using linear-time document clustering. *KDD '99*.

Gina-Anne Levow, 2006. Unsupervised and semi-supervised learning of tone and pitch accent. *HLT-NAACL '06*.

Marina Meila and David Heckerman, 2001. An experimental comparison of model-based clustering methods. *Machine Learning*, 42(1/2):9-29.

Marina Meila, 2007. Comparing clustering – an information based distance. *Journal of Multivariate Analysis*, 98:873–895.

C.W Milligan, S.C Soon and L.M Sokol, 1983. The effect of cluster size, dimensionality and the number of clusters on recovery of true cluster structure. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 5:40–47.

Boris G. Mirkin, 1996. Mathematical classification and clustering. *Kluwer Academic Press.*

Darius M. Pfitzner, Richard E. Leibbrandt and David M.W Powers, 2008. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowledge and Information Systems: An International Journal*, DOI 10.1007/s10115-008-0150-6.

William Rand, 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statstical Association*, 66(336):846–850.

Roi Reichart and Ari Rappoport, 2008. Unsupervised induction of labeled parse trees by clustering with syntactic features. *COLING '08*.

Andrew Rosenberg and Julia Hirschberg, 2007. V–Measure: a conditional entropy–based external cluster evaluation measure. *EMNLP '07*.

Sabine Schulte im Walde, 2003. Experiments on the automatic induction of German semantic verb classes. *Ph.D. thesis, Universitat Stuttgart*.

Yoav Seginer, 2007. Fast unsupervised incremental parsing. *ACL 07*.

Sa-Im Shin and Key-Sun Choi, 2004. Automatic word sense clustering using collocation for sense adaptation. *The Second Global WordNet Conference*.

Stijn van Dongen, 2000. Performance criteria for graph clustering and markov cluster experiments. *Technical report CWI, Amsterdam*

Daniel D. Walker and Eric K. Ringger, 2008. Model-based document clustering with a collapsed Gibbs sampler. *KDD '08*.

Nianwen Xue, Fu-Dong Chiou and Martha Palmer, 2002. Building a large–scale annotated Chinese corpus. *ACL '02*.

Yujing Zeng, Jianshan Tang, Javier Garcia-Frias, and Guang R. Gao, 2002. An adaptive meta-clustering approach: combining the information from different clustering results. *IEEE Computer Society Bioinformatics Conference (CSB '02)* .

Ying Zhao and George Karypis, 2001. Criterion functions for document clustering: experiments and analysis. *Technical Report TR 01-40, Department of Computer Science, University of Minnesota.*

Lexical Patterns or Dependency Patterns: Which Is Better for Hypernym Extraction?

Erik Tjong Kim Sang
Alfa-informatica
University of Groningen
e.f.tjong.kim.sang@rug.nl

Katja Hofmann
ISLA, Informatics Institute
University of Amsterdam
khofmann@science.uva.nl

Abstract

We compare two different types of extraction patterns for automatically deriving semantic information from text: lexical patterns, built from words and word class information, and dependency patterns with syntactic information obtained from a full parser. We are particularly interested in whether the richer linguistic information provided by a parser allows for a better performance of subsequent information extraction work. We evaluate automatic extraction of hypernym information from text and conclude that the application of dependency patterns does not lead to substantially higher precision and recall scores than using lexical patterns.

1 Introduction

For almost a decade, automatic sentence parsing systems with a reasonable performance (90+% constituent precision/recall) have been available for English (Charniak, 1999). In recent years there has been an increase in linguistic applications which use parsing as a preprocessing step, e.g. Snow et al. (2006) and Surdeanu et al. (2008). One of the boosts for these new applications was the increasing power of desktop computers, which allows for an easier access to the computing-intensive parsing results. Another is the increased popularity of dependency parsing of which the results can easily be incorporated into followup systems.

Although there is a consensus about the fact that the richness of the dependency structures should, in principle, enable better performance than lexical information or shallow parsing results, it is not clear if these better results can also be obtained in practice. A performance of 90% precision and recall at constituent level still leaves an average of one error in a medium-length sentence of ten words. These errors could degrade the performance of any approach which relies heavily on parser output.

The question of whether to include a full parser as a preprocessor for natural language processing task, has led to a heated discussion between the two authors of the paper. One of us argues that full parsers are slow and make too many errors, and relies on shallow techniques like part-of-speech tagging for preprocessing. The other points at the decreasing costs of computing and improvements in the reliability of parsers, and recommends dependency parsers as preprocessing tools.

While no automatic text preprocessing method is free of errors, it is indeed true that approaches other than full parsing, like for example shallow parsing, offer useful information at a considerably cheaper processing cost. The choice between using a heavy full parser or a light shallow language analyzer is one that developers of language processing systems frequently have to make. The expected performance boost of parsed data could be an important motivation for choosing for full syntactic analysis. However, we do not know how big the difference between the two methods will be. In order to find this out, we designed an experiment in which we compared the effects of preprocessing with and without using information generated by a full parser.

In this paper, we compare two text preprocessing approaches for a single language processing task. The first of the two methods is shallow lin-

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 174–182,
Boulder, Colorado, June 2009.

guistic processing, a robust and fast text analysis method which only uses information from words, like lemma information and part-of-speech classes. The second method is dependency parsing which includes information about the syntactic relations between words. The natural language processing task which we will use for assessing the usability of the two processing methods is automatic extraction of hypernym information from text. The language of the text documents is Dutch. We expect that the findings of this study would have been similar if any other Germanic language (including English) was used.

The contribution of this paper is a thorough and fair comparison of the involved preprocessing techniques. There have been earlier studies of hypernym extraction with either lexical or dependency extraction patterns. However, these studies applied the techniques to a variety of different data sets and used different evaluation techniques. We will apply the two methods to the same data, evaluate the results in a consistent manner and examine the differences.

After this introduction, we will describe the task, the preprocessing methods and the evaluation setting in more detail. In the third section, we will show how our experiments were set up and present the results. Section four contains a detailed discussion of the two methods and their effect on the extraction task. In the final section of the paper, we will present some concluding remarks.

2 Task and methods

We will apply two different preprocessing methods to the task of extracting lexical information from text. In the next sections we describe this task, discuss different methods for preprocessing the data and outline the method used for evaluating the results.

2.1 Extracting hypernym relations

We will concentrate on extracting a single type of lexical relation: hypernymy. Word A is a hypernym of word B if the meaning of A both covers the meaning of B and is broader. For example, *color* is a hypernym of *red* which in turn is a hypernym of *scarlet*. If A is a hypernym of B than B is a hyponym of A.

There has been quite a lot of work on extracting hypernymy pairs from text. The pioneering work of Hearst (1992) applied fixed patterns like NP_1 , *especially* NP_2 to derive that NP_1 is a hypernym of NP_2. Lately there has been a lot of interest in acquiring such text patterns using a set of hypernymy examples, e.g. Pantel et al. (2004) and Snow (2006). Application of such techniques has not been restricted to English but also involved other languages such as Dutch (Tjong Kim Sang and Hofmann, 2007). Recent work has also examined extracting hypernym information from structured data, like Wikipedia (Sumida and Torisawa, 2008).

For our extraction work, we will closely follow the approach described in Snow et al. (2006):

1. Collect from a text corpus phrases (consecutive word sequences from a single sentence) that contain a pair of nouns
2. Mark each phrase as containing a hypernym pair or a non-hypernym pair according to a lexical resource
3. Remove the noun pair from the phrases and register how often each phrase is associated by hypernym pairs and by non-hypernym pairs
4. Use this information for training a machine learning system to predict whether two nouns are a hypernym-hyponym pair based on the phrases in which they occur in a text corpus

For example, we find two phrases: *colors such as cyan* and *colors such as birds*, both of which contain the basic phrase *such as*. We mark the first phrase as a hypernym phrase (*color* is a hypernym of *cyan*) while the second is marked as non-hypernym (*color* is not a hypernym of *bird*). Thus the pattern *such as* will receive a positive point and a negative point. A machine learning algorithm can deduce from these numbers that two other nouns occurring in the same pattern will have an estimated probability of 50% of being related according to hypernymy. The learner can use information from other patterns to obtain a better estimation of this probability.

2.2 Lexical patterns

We use two different text preprocessing methods which automatically assign linguistic information to sentences. The first preprocessing method has the

advantage of offering a fast analysis of the data but its results are less elaborate than those of the second method. The first method consists of three steps:

- Tokenization: sentence boundaries are detected and punctuation signs are separated from words
- Part-of-speech tagging: part-of-speech classes like noun and verb are assigned to words
- Lemmatization: words are reduced to their basic form (lemma)

The analysis process would convert a phrase like *Large cities in northern England such as Liverpool are beyond revival.* to lemmas and their associated part-of-speech tags: *large*/JJ *city*/NN *in*/IN *north*/JJ *England*/NNP *such*/DT *as*/IN *Liverpool*/NNP *be*/VB *beyond*/IN *revival*/NN ./.

Like in the work of Snow et al. (2005), the target phrases for hypernym extraction are two noun phrases, with a maximum of three tokens in between and one or two optional extra tokens (a non-head token of the first noun phrase and/or one of the second noun phrase). The lexical preprocessing method uses two basic regular expressions for finding noun phrases: *Determiner? Adjective* Noun+* and *ProperNoun+*. It assumes that the final token of the matched phrase is the head. Here is one set of four patterns which can be derived from the example sentence:

1. *NP in NP*
2. *large NP in NP*
3. *NP in north NP*
4. *large NP in north NP*

The patterns contain the lemmas rather than the words of the sentence in order to allow for general patterns. For the same reason, the noun phrases have been replaced by the token NP. Each of the four patterns will be used as evidence for a possible hypernymy relation between the two noun phrase heads *city* and *England*. As a novel extension to the work of Snow et al., we included two additional variants of each pattern in which either the first NP or the second NP was replaced by its head:

5. *city in NP*
6. *NP in England*

This enabled us to identify among others appositions as patterns: *president NP*.

2.3 Dependency patterns

A dependency analysis contains the same three steps used for finding lexical patterns: tokenization, part-of-speech tagging and lemmatization. Additionally, it includes a fourth step:

- Dependency parsing: find the syntactic dependency relations between the words in each sentence

The syntactic analysis is head-based which means that for each word in the sentence it finds another word that dominates it. Here is a possible analysis of the previous example sentence:

large:JJ:MOD:NN:*city*
city:NN:SUBJ:VBD:*be*
in:IN:MOD:NN:*city*
north:JJ:MOD:NNP:*England*
England:NNP:OBJ1:IN:*in*
such:DT:MOD:IN:*as*
as:IN:MOD:NN:*city*
Liverpool:NNP:OBJ1:IN:*as*
be:VB:–:–:–
beyond:IN:MOD:VB:*be*
revival:NN::OBJ1:IN:*beyond*

Each line contains a lemma, its part-of-speech tag, the relation between the word and its head, the part-of-speech tag of its head and the lemma of the head word. Our work with dependency patterns closely follows the work of Snow et al. (2005). Patterns are defined as dependency paths with at most three intermediate nodes between the two focus nouns. Additional satellite nodes can be present next to the two nouns. The dependency patterns contain more information than the lexical patterns. Here is one of the patterns that can be derived for the two noun phrases *large cities* and *northern England* in the example sentence:

NP_1:NN:SUBJ:VBD:
in:IN:MOD:NN:NP_1
NP_2:NNP:OBJ1:IN:*in*

The pattern defines a path from the head lemma *city* via *in*, to *England*. Note that lemma information linking outside this pattern (*be* at the end of the first line) has been removed and that lemma information

from the target noun phrases has been replaced by the name of the noun phrase (NP_1 at the end of the second line). For each dependency pattern, we build six variants similar to the six variants of the lexical patterns: four with additional information from the two noun phrases and two more with head information of one of the two target NPs.

Both preprocessing methods can identify phrases like *N such as* N_1 *,* N_2 *and* N_3 as well. Such phrases produce evidence for each of the pairs (N,N_1), (N,N_2) and (N,N_3). These three noun pairs will be included in the data collected for the patterns that can be derived from the phrase.

We expect that an important advantage of using dependency patterns over lexical patterns will be that the former offer a wider coverage. In the example sentence, no lexical pattern will associate *city* with *Liverpool* because there are too many words in between. However, a dependency pattern will create a link between these two words, via the word *as*. This will enable the dependency patterns to find out that *city* is a hypernym of *Liverpool*, where the lexical patterns are not able to do this based on the available information.

The two preprocessing methods generate a large number of noun pairs associated by patterns. Like Snow et al. (2005), we keep only noun pairs which are associated by at least five different patterns. The same constraint is enforced on the extraction patterns: we keep only the patterns which are associated by at least five different noun pairs. The data is converted to binary feature vectors representing noun pairs. These are training data for a Bayesian Logistic Regression system, BBRtrain (Genkin et al., 2004). We use the default settings of the learning system and test its prediction capability in a binary classification task: whether two nouns are related according to hypernymy or not. Evaluation is performed by 10-fold cross validation.

2.4 Evaluation

For parameter optimization we need an automatic evaluation procedure, since repeated manual checks of results generated by different versions of the learner require too much time. We have adopted the evaluation method of Snow et al (2006): compare the generated hypernyms with hypernyms present in a lexical resource, in our case the Dutch part of EuroWordNet (1998).

This choice results in two restrictions. First, we will only consider pairs of known words (words that are present in the lexical resource) for evaluation. We have no information about other words so we make no assumptions about them. Second, if two words appear in the lexical resource but not in the hypernym relation of that same resource then we will assume that they are unrelated. In other words, we assume the hypernymy relation specified in the lexical resource as complete (like in the work of Snow et al. (2006)).

We use standard evaluation scores. We will compute precision and recall for the candidate hypernyms, as well as the related $F_{\beta=1}$ rate, the harmonic mean between precision and recall. Precision will be computed against all chosen candidate hypernyms. However, recall will only be computed against the positive noun pairs which occur in the phrases selected by the examined method. The different preprocessing methods may cause different numbers of positive pairs to be selected. Only these pairs will be used for computing recall scores. Others will be ignored. For this reason we will report the selected number of positive target pairs in the result tables as well[1].

3 Experiments and results

We have applied the extraction techniques to two different Dutch corpora. The first is a collection of texts from the news domain. It consists of texts from five different Dutch news papers from the Twente News Corpus collection. Two versions of this corpus exist. We have worked with the version which contains the years 1997-2005 (26 million sentences and 450 million tokens). The second corpus is the Dutch Wikipedia. Here we used a version of October 2006 (5 million sentences and 58 million words).

Syntactic preprocessing of the material was done with the Alpino parser, the best available parser for Dutch with a labeled dependency accuracy of 89% (Van Noord, 2006). Rather than performing the parsing task ourselves, we have relied on an available parsed treebank which included the text corpora

[1] In a seperate study we have shown that the observed differences between the two methods remain the same when recall is computed over sets of similar sizes (Tjong Kim Sang, 2009).

that we wanted to use (Van Noord, 2009).

The parser also performs part-of-speech tagging and lemmatization, tasks which are useful for the lexical preprocessing methods. However, taking future real-time applications in mind, we did not want the lexical processing to be dependent on the parser. Therefore we have developed an in-house part-of-speech tagger and lemmatizer based on the material created in the Corpus Spoken Dutch project (Eynde, 2005). The tagger achieved an accuracy of 96% on test data from the same project while the lemmatizer achieved 98%.

We used the Dutch part of EuroWordNet (Vossen, 1998) as the gold standard lexical resource, both for training and testing. In the lexicon, many nouns have different senses. This can cause problems for the pattern extraction process. For example, if a noun N_1 with sense X is related to another noun N_2 then the appearance of N_1 with sense Y with N_2 in the text may be completely accidental and say nothing about the relation between the two words. In that case it would be wrong to regard the context of the two words as an interesting extraction pattern.

There are several ways to deal with this problem. One is to automatically assign senses to words. However we do not have a reliable sense tagger for Dutch at our disposal. Another method was proposed by Snow et al (2005): assume that every word bears its most frequent sense. But this is also information which we lack for Dutch: our lexical resource does not contain frequency information for word senses. We have chosen the approach suggested by Hofmann and Tjong Kim Sang (2007): remove all nouns with multiple senses from the data set and use only the monosemous words for finding good extraction patterns. This restriction is only imposed in the training phase. We consider both monosemous words and polysemous words in the evaluation process.

We imposed two additional restrictions on the lexical resource. First, we removed the top noun of the hypernymy hierarchy (*iets*) from the list of valid hypernyms. This word is a valid hypernym of any other noun. It is not an interesting suggestion for the extraction procedure to put forward. Second, we restricted the extraction procedure to propose only known hypernyms as candidate hypernyms. Nouns that appeared in the lexical resources only as hyponyms (leaf nodes of the hypernymy tree) were never proposed as candidate hypernyms. This made sense for our evaluation procedure which is only aimed at finding known hypernym-hyponym pairs.

lexical patterns

Data source	Targ.	Prec.	Recall	$F_{\beta=1}$
AD	620	55.8%	27.9%	37.2
NRC	882	50.4%	23.8%	32.3
Parool	462	51.8%	21.9%	30.8
Trouw	607	54.1%	25.9%	35.0
Volkskrant	970	49.7%	24.1%	32.5
Newspapers	3307	43.1%	26.7%	33.0
Wikipedia	1288	63.4%	44.3%	52.1

dependency patterns

Data source	Targ.	Prec.	Recall	$F_{\beta=1}$
AD	706	42.9%	30.2%	35.4
NRC	1224	26.2%	25.3%	25.7
Parool	584	31.2%	23.8%	27.0
Trouw	760	35.3%	29.0%	31.8
Volkskrant	1204	29.2%	25.5%	27.2
Newspapers	3806	20.7%	29.1%	24.2
Wikipedia	1580	61.9%	47.0%	53.4

Table 1: Hypernym extraction scores for the five newspapers in the Twente News Corpus (AD, NRC, Parool, Trouw and Volkskrant) and for the Dutch Wikipedia. The Targets column shows the number of unique positive word pairs in each data set. The Dutch Wikipedia contains about as much data as one of the newspaper sections.

We performed two hypernym extraction experiments, one which used lexical extraction patterns and one which used dependency patterns[2]. The results from the experiments can be found in Table 1. The newspaper F-scores obtained with lexical patterns are similar to those reported for English (Snow et al., 2005, 32.0) but the dependency patterns perform worse. Both approaches perform well on Wikipedia data, most likely because of the more repeated sentence structures and the presence of many definition sentences. For newspaper data, lexical patterns outperform dependency patterns both for precision and $F_{\beta=1}$. For Wikipedia data the differences are smaller and in fact the dependency pat-

[2]The software used in these experiment has been made available at http://www.let.rug.nl/erikt/cornetto/D08.zip

terns obtain the best F-score. For all data sets, the dependency patterns suggest more related pairs than the lexical patterns (column Targets). The differences between the two pattern types are significant ($p < 0.05$) for all evaluation measures for Newspapers and for positive targets and recall for Wikipedia.

4 Result analysis

In this section, we take a closer look at the results described in the previous section. We start with looking for an explanation for the differences between the scores obtained with lexical patterns and dependency patterns. First we examine the results for Wikipedia data and then the results for newspaper data. Finally, we perform an error analysis to find out the strengths and weaknesses of each of the two methods.

4.1 Wikipedia data

The most important difference between the two pattern types for Wikipedia data is the number of positive targets (Table 1). Dependency patterns find 23% more related pairs in the Wikipedia data than lexical patterns (1580 vs. 1288). This effect can also be simulated by changing the size of the corpus. If we restrict the data set of the dependency patterns to 70% of its current size then the patterns retrieve a similar number of positive targets as the lexical patterns, 1289, with comparable precision, recall and $F_{\beta=1}$ scores (62.5%, 46.6% and 53.4). So we expect that the effect of applying the dependency patterns is the same as applying the lexical patterns to 43% more data.

4.2 Newspaper data

Performance-wise there seems to be only a small difference between the two preprocessing methods when applied to the Wikipedia data set. However, when we examine the scores obtained on the newspaper data (Table 1) then we find larger differences. Dependency patterns remain finding more positive targets and obtaining a larger recall score but their precision score is disappointing. However, when we examine the precision-recall plots of the two methods (Figure 1, obtained by varying the acceptance threshold of the machine learner), they are almost indistinguishable. The performance line for lexical patterns extends further to the left than the one of

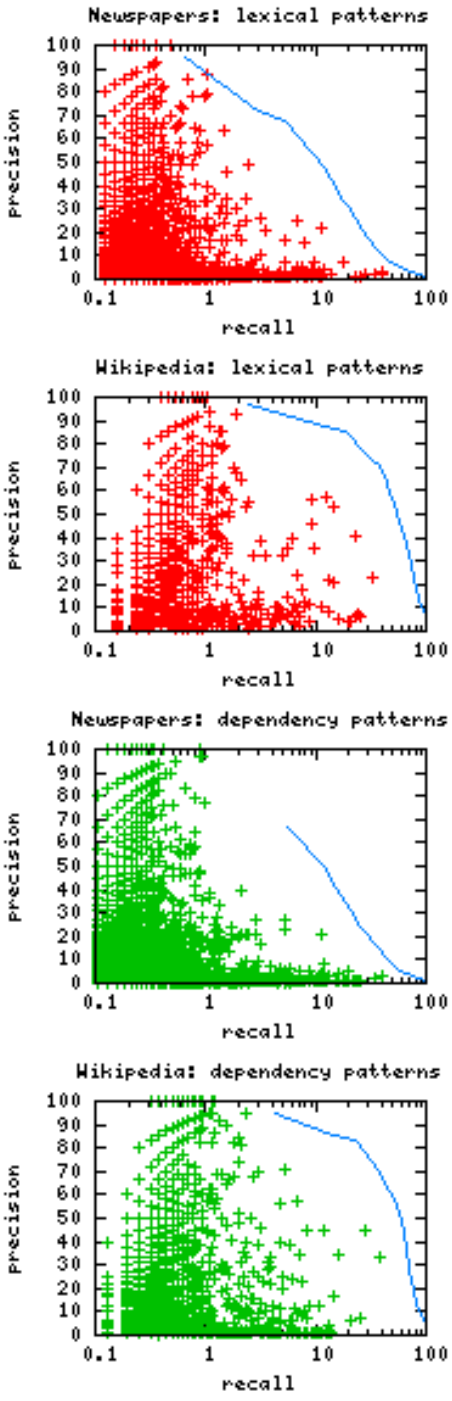


Figure 1: Performance of individual hypernym extraction patterns applied to the combination of five newspapers and Wikipedia. Each + in the graphs represent a different extraction pattern. The precision-recall graphs for the machine learner (lines) are identical for each data source except for the extended part of the performance line for lexical patterns.

lexical patterns applied to Newspapers

Key Phrase	Targ.	Prec.	Recall	$F_{\beta=1}$
N and other N	376	22.0%	11.4%	15.0
N such as N	222	25.1%	6.7%	10.6
N like N	579	7.6%	17.5%	10.6
N , such as N	263	15.6%	8.0%	10.5
N (N	323	7.5%	9.8%	8.5

dependency patterns applied to Newspapers

Key Phrase	Targ.	Prec.	Recall	$F_{\beta=1}$
N and other N	420	21.1%	11.0%	14.5
N be a N	451	8.2%	11.8%	9.7
N like N	205	27.3%	5.4%	9.0
N be N	766	5.7%	20.1%	8.8
N such as N	199	22.4%	5.2%	8.5

lexical patterns applied to Wikipedia

Key Phrase	Targ.	Prec.	Recall	$F_{\beta=1}$
N be a N	294	40.8%	22.8%	29.3
N be N	418	22.9%	32.5%	26.9
a N be N	185	53.3%	14.4%	22.6
N such as N	161	57.5%	12.5%	20.5
N (N	188	21.2%	14.6%	17.3

dependency patterns applied to Wikipedia

Key Phrase	Targ.	Prec.	Recall	$F_{\beta=1}$
N be N	609	33.6%	38.5%	35.9
N be a N	452	44.3%	28.6%	34.8
the N be N	258	34.0%	16.3%	22.1
a N be N	184	44.7%	11.6%	18.5
N N	234	16.6%	14.8%	15.6

Table 2: Best performing extraction patterns according to F-scores.

the dependency patterns but the remainder of the two graphs overlap. The measured performances in Table 1 are different because the machine learner put the acceptance level for extracted pairs at different points of the graph: the performance lines in both newspaper graphs contain (recall,precision) points (26.7%,43.1%) and (29.1%,20.7%).

We are unable to find major differences in the results of the two approaches. We conclude that, apart from an effect which can be simulated with some extra data, there is no difference between preprocessing text with shallow methods and with a full dependency parser.

56	—	covered by other patterns
12	48%	required full parsing
6	24%	lemmatization errors
3	12%	omitted for lack of support
3	12%	pos tagging errors
1	4%	extraction pattern error
81	100%	

45	—	covered by other patterns
38	64%	parsing errors
10	17%	lemmatization errors
7	12%	extraction pattern errors
3	5%	omitted for lack of support
1	2%	pos tagging error
104	100%	

Table 3: Primary causes of recall errors made by the lexical pattern *N such as N* (top) and the best performing corresponding dependency pattern (bottom).

4.3 Error analysis

Despite the lack of performance differences between the two preprocessing methods, there are still internal differences which cause one method to generate different related word pairs than the other. We will now examine in detail two extraction patterns and specify their distinct effects on the output results. We hope that by carefully examining their output we can learn about the strengths and weaknesses of the two approaches.

We take a closer look at extraction pattern *N such as N* for Newspaper data (second best for lexical patterns and fifth best for dependency patterns, see Table 2). The lexical pattern found 222 related word pairs while the dependency pattern discovered 199. 118 of these pairs were found by both patterns which means that the lexical pattern missed 81 of the pairs while the dependency pattern missed 104.

An overview of the cause of the recall errors can be found in Table 3. The two extraction patterns do not overlap completely. The dependency parser ignored punctuation signs and therefore the dependency pattern covers both phrases with and without punctuation. However, these phrase variants result in different lexical patterns. This is the cause for 56 hypernyms being missed by the lexical pattern.

Meanwhile there is a difference between a dependency pattern without the conjunction *and* and one with the conjunction, while there is a unified lexical pattern processing both phrases with and without conjunctions. This caused the dependency pattern to miss 45 hypernyms. However, all of these 'missed' hypernyms are handled by other patterns.

The main cause of the recall differences between the two extraction patterns was the parser. The dependency pattern found twelve hypernyms which the lexical pattern missed because they required an analysis which was beyond part-of-speech tagging and the basic noun phrase identifier used by the lexical preprocessor. Six hypernyms required extending a noun phrase with a prepositional phrase, five needed noun phrase extension with a relative clause and one involved appositions. An example of such a phrase is *illnesses caused by vitamin deficits, like scurvy and beriberi*.

However, the syntactic information that was available to the dependency pattern did also have a negative effect on its recall. 38 of the hypernyms detected by the lexical pattern were missed by the dependency pattern because there was a parsing error in the relevant phrase. In more than half of the cases, this involved attaching the phrase starting with *such as* at an incorrect position. We found that a phrase like N_1 *such as* N_2 *,* N_3 *and* N_4 could have been split at any position. We even found some cases of prepositional phrases and relative clauses incorrectly being moved from other positions in the sentence into the target phrase.

Other recall error causes appear less frequently. The two preprocessing methods used different lemmatization algorithms which also made different errors. The effects of this were visible in the errors made by the two patterns. Some hypernyms that were found by both patterns but were not present in both results because of insufficient support from other patterns (candidate hypernyms should be supported by at least five different patterns). The effect of errors in part-of-speech tags was small. Our data analysis also revealed some inconsistencies in the extraction patterns which should be examined.

5 Concluding remarks

We have evaluated the effects of two different preprocessing methods for a natural language processing task: automatically identifying hypernymy information. The first method used lexical patterns and relied on shallow processing techniques like part-of-speech tagging and lemmatization. The second method used dependency patterns which relied on additional information obtained from dependency parsing.

In earlier work, McCarthy et al. (2007) found that for word sense disambiguation using thesauri generated from dependency relations perform only slightly better than thesauri generated from proximity-based relations. Jijkoun et al. (2004) showed that information obtained from dependency patterns significantly improved the performance of a question answering system. Li and Roth (2001) report that preprocessing by shallow parsing allows for a more accurate post-processing of ill-formed sentences than preprocessing with full parsing.

Our study supports the findings of McCarthy et al. (2007). We found only minor differences in performances between the two preprocessing methods. The most important difference: about 20% extra positive cases that were identified by the dependency patterns applied to Wikipedia data, can be overcome by increasing the data set of the lexical patterns by half. We believe that obtaining more data may often be easier than dealing with the extra computing time required for parsing the data. For example, in the course of writing this paper, we had to refrain from using a recent version of Wikipedia because parsing the data would have taken 296 *days* on a single processor machine compared with a single hour for tagging the data.

References

Eugene Charniak. 1999. A maximum-entropy inspired parser. Technical Report CS-99-12, Brown University.

Frank Van Eynde. 2005. *Part of Speech Tagging en Lemmatisering van het Corpus Gesproken Nederlands*. K.U. Leuven. (in Dutch).

Alexander Genkin, David D. Lewis, and David Madigan. 2004. *Large-Scale Bayesian Logistic Regression for Text Categorization*. Technical report, Rutgers University, New Jersey.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of ACL-92*. Newark, Delaware, USA.

Katja Hofmann and Erik Tjong Kim Sang. 2007. Automatic extension of non-english wordnets. In *Proceedings of SIGIR'07*. Amsterdam, The Netherlands (poster).

Valentin Jijkoun, Maarten de Rijke, and Jori Mur. 2004. Information extraction for question answering: Improving recall through syntactic patterns. In *Proceedings of Coling'04*. Geneva, Switzerland.

Xin Li and Dan Roth. 2001. Exploring evidence for shallow parsing.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Caroll. 2007. Unsupervised acquisition of predominant word senses. *Computational Linguistics*, 33(4).

Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. 2004. Towards terascale knowledge acquisition. In *Proceedings of COLING 2004*, pages 771–777. Geneva, Switzerland.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS 2005*. Vancouver, Canada.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING/ACL 2006*. Sydney, Australia.

Asuka Sumida and Kentaro Torisawa. 2008. Hacking wikipedia for hyponymy relation acquisition. In *Proceedings of IJCNLP 2008*. Hyderabad, India.

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

Erik Tjong Kim Sang and Katja Hofmann. 2007. Automatic extraction of dutch hypernym-hyponym pairs. In *Proceedings of CLIN-2006*. Leuven, Belgium.

Erik Tjong Kim Sang. 2009. To use a treebank or not – which is better for hypernym extraction. In *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT 7)*. Groningen, The Netherlands.

Gertjan Van Noord. 2006. At last parsing is now operational. In Piet Mertens, Cedrick Fairon, Anne Dister, and Patrick Watrin, editors, *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*.

Gertjan Van Noord. 2009. Huge parsed corpora in lassy. In *Proceedings of TLT7*. LOT, Groningen, The Netherlands.

Piek Vossen. 1998. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publisher.

Improving Text Classification by a Sense Spectrum Approach to Term Expansion

Peter Wittek
Department of Computer Science
National University of Singapore
Computing 1, Law Link
Singapore 117590
wittek@comp.nus.edu.sg

Sándor Darányi
Swedish School of Library
and Information Science
Göteborg University &
University of Borås
Allégatan 1
50190 Borås, Sweden
sandor.daranyi@hb.se

Chew Lim Tan
Department of Computer Science
National University of Singapore
Computing 1, Law Link
Singapore 117590
tancl@comp.nus.edu.sg

Abstract

Experimenting with different mathematical objects for text representation is an important step of building text classification models. In order to be efficient, such objects of a formal model, like vectors, have to reasonably reproduce language-related phenomena such as word meaning inherent in index terms. We introduce an algorithm for sense-based semantic ordering of index terms which approximates Cruse's description of a sense spectrum. Following semantic ordering, text classification by support vector machines can benefit from semantic smoothing kernels that regard semantic relations among index terms while computing document similarity. Adding expansion terms to the vector representation can also improve effectiveness. This paper proposes a new kernel which discounts less important expansion terms based on lexical relatedness.

1 Introduction

Generally, building an automated text classification system consists of two key subtasks. The first task is text representation which converts the content of documents into compact format so that they can be further processed by the text classifiers. Another task is to learn the model of a text classifier which is used to classify the unlabeled documents. This paper proposes a substantially new model for text representation to improve effectiveness of text classification by semantic ordering.

Our motivation for the research presented here came from (Dorrer et al., 2001) who demonstrated the viability of database searching by visible light using a quantum algorithm, albeit on meaningless items. The question was, what kind of document representation would be necessary to extend their in-principle results to include semantics, one that has been leading us to test both periodic and non-periodic functions for this purpose. Since representation and retrieval by colors was implied in their method, we speculated that the following components could be useful in a rephrased model: (a) a metaphorically presented spectral expression of lexical semantic phenomena, (b) a ranked one-dimensional condensate of multidimensional sense structure, and (c) representation of documents and queries by functions in L_2 space with a similarity measure. Our anticipation was that by matching these components, a new model could demonstrate new capacities in general, and contribute to computing meaning by waves in particular.

Semantic ordering (component b) is an approximation of what (Cruse, 1986) referred to as a sense spectrum, i.e. a series of points - called local senses and constituting lexical units -, in a one-dimensional semantic continuum (component a). Apart from differentiating between the conceptual content of the same word in terms of its senses in word pairs, i.e. their semantic relatedness, it also compresses the result in spectral form. The scalar values of this spectrum have the double potential of being a condensed measure for semantic weighting, and, tentatively, they can play the role of mass in experiments where gravity is called in as a metaphor for text categorization and information retrieval (Paijmans, 1997; Shi et al., 2005; Wittek et al., 2009).

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 183–191,
Boulder, Colorado, June 2009.

This paper addresses text categorization by means of non-periodical functions only.

In support of Cruse's point, recently it has been demonstrated by measurements that sense classification errors made by their maximum entropy based word sense disambiguation system were partly remedied once instead of a fine-grained view, a more coarse-grained view of senses was adopted (Palmer et al., 2006). Improvement of sense classification accuracy linked with "zooming out" in terms of observation granularity indicates, in our eyes, the "fluid", perhaps spectral nature of sense inasmuch as it is impossible to precisely distinguish between the borderlines and some fuzziness is implied both in the phenomenon and its perception. This "fluidity of language", as Palmer et al. call it, is in accord with the theory of shared semantic representations in psycholinguistics (Rodd et al., 2002), according to which related senses share a portion of their meaning representation in the mental lexicon; it also supports an earlier observation of two of the present authors based on the same methodology as outlined in this paper, namely that using continuous functions for information retrieval leads to content representation without exact term or document locations, one which is regional in its nature and subject to a mathematical uncertainty principle (Wittek and Darányi, 2007).

We approach our problem in three steps: (1) whether distributional semantics alone is enough for the representation of word meaning, (2) whether semantic relatedness between word pairs can be expressed in an ordered form while preserving lexical field structure, and if (3) the uniqueness of entries in such an order can be expressed by functions rather than scalars such as distance. As we will show, this line of thought leads to performance improvement in text classification by using kernel-based feature weighting.

Since the early days of the vector space model, it has been debated whether it is a proper carrier of meaning of texts (Raghavan and Wong, 1986), arguing if distributional similarity is an adequate proxy for lexical semantic relatedness (Budanitsky and Hirst, 2006). We argue for the need to enrich distributional semantics-based text representation by other components because with the statistical, i.e. devoid of word semantics approaches there is generally no way to improve both precision and recall at the same time, increasing one is done at the expense of the other. For example, casting a wider net of search terms to improve recall of relevant items will also bring in an even greater proportion of irrelevant items, lowering precision. In the meantime, practical approaches have been proliferating, especially with developments in kernel methods in the last decade (Joachims, 1998; Cristianini et al., 2002). Some researchers suggested a more general mathematical framework to accommodate the needs that the vector space model cannot satisfy (van Rijsbergen, 2004). This paper explores the opportunities of this representation in the domain of text classification by introducing it as a new nonlinear semantic kernel.

Another aspect of the same problem is term expansion for document classification and retrieval. By automatically selecting expansion terms for a text classification system to expand a document vector by adding terms that are related to the terms already in the document, performance can be improved (Hu et al., 2008). Such new terms can either be statistically related to the original terms or chosen from lexical resources such as thesauri, controlled vocabularies, ontologies and the like.

However, in doing so the fundamental question often overlooked is whether the expansion terms extracted are equally related to the document and are useful for text classification. In what follows we propose a form of term expansion with decreasing importance of those terms that are less related, as contrasted with rigid term expansion. This can be carried out by a combination of semantic ordering and using function space for classification.

This paper is organized as follows. Section 2 overviews text classification by support vector machines, expanding on traditional text similarity measures (Section 2.1), semantic smoothing kernels (Section 2.2), term expansion strategies (Section 2.3), and finally introduces our semantic kernels in the L_2 space (Section 2.4). Section 3 discusses experimental results and Section 4 concludes the paper.

2 Text Classification with Support Vector Machines

Text categorization is the task of assigning unlabeled documents into predefined categories. Given a collection of $\{d_1, d_2, \ldots, d_N\}$ documents, and a $C = \{c_1, c_2, ..., c_{|C|}\}$ set of predefined categories, the task is, for each document d_j ($j \in \{1, 2, \ldots, N\}$), to assign a decision to file d_j under c_i or a decision not to file d_j under c_i ($c_i \in C$) by virtue of a function Φ, where the function Φ is also referred to as the classifier, or model, or hypothesis, or rule. Supervised text classification is a machine learning technique for creating the function Φ from training data. The training data consist of pairs of input documents, and desired outputs (i.e., classes).

Support vector machines have been found the most effective by several authors (Joachims, 1998). The proposed semantic text classification method is grounded in the kernel methods underlying support vector machines.

A support vector machine is a kind of supervised learning algorithm. In its simplest, linear form, a support vector machine is a hyperplane that separates a set of positive examples from a set of negative examples with maximum margin (Shawe-Taylor and Cristianini, 2004). The strength of kernel methods is that they allow a mapping $\phi(.)$ of $\mathbf{x}$ to a higher dimensional space. In the dual formulation of the mathematical programming problem, only the kernel matrix $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)'\phi(\mathbf{x}_i)$ is needed in the calculations.

2.1 Traditional Text Similarity Measure

Intuitively, if a text fragment of two documents address similar topics, it is highly possible that they share lots of substantive terms. After having removed the stopwords and stemmed the rest, the stemmed terms construct a vector representation for each text document. Let $\mathbf{a}_j$ be a document vector in the vector space model, that is, $\mathbf{a}_j = \sum_{k=1}^{M} a_{kj}\mathbf{e}_k$, where M is the number of index terms, a_{kj} is some weighting (e.g., term frequency), and $\mathbf{e}_k$ is a basis vector of the M-dimensional Euclidean space. This representation is also referred to as the bag-of-words (BOW) model.

Given this representation, semantic relatedness of a pair of text fragments is computed as the cosine similarity of their corresponding term vectors which is defined as:

$$S(\mathbf{a}_i, \mathbf{a}_j) = \frac{\mathbf{a}_i \mathbf{a}_j}{|\mathbf{a}_i||\mathbf{a}|_j}. \quad (1)$$

2.2 Linear Semantic Kernels

One enrichment strategy is to use a semantic smoothing kernel while calculating the similarity between two documents. Any linear kernel for texts is characterized by $K(\mathbf{a}_i, \mathbf{a}_j) = \mathbf{a}_i' S' S \mathbf{a}_j$, where S is an appropriately shaped matrix commonly referred to as semantic smoothing matrix (Siolas and d'Alché Buc, 2000; Shawe-Taylor and Cristianini, 2004; Basili et al., 2005; Mavroeidis et al., 2005; Bloehdorn et al., 2006). The presence of S changes the orthogonality of the vector space model, as this mapping should introduce term dependence. A recent attempt tried to manually construct S with the help of a lexical resource (Siolas and d'Alché Buc, 2000). The entries in the symmetric matrix S express the semantic similarity between the terms i and j. Entries in this matrix are inversely proportional to the length of the WordNet hierarchy path linking the two terms. The performance, measured over the 20NewsGroups corpus, showed an improvement of 2 % over the the basic vector space method. Moreover, the semantic matrix S is almost fully dense, hence computational issues arise. In a similar construction, (Bloehdorn et al., 2006) defined the matrix entries as weights of superconcepts of the two terms in the WordNet hierarchy. Focusing on special subcategories of Reuters-21578 and on the TREC Question Answering Dataset, they showed consistent improvement over the baseline. As (Mavroeidis et al., 2005) pointed out, polysemy will remain a problem in semantic smoothing kernels. A more complex way of calculating the semantic similarity as the matrix entries was also proposed (Basili et al., 2005). For a more general discussion on semantic similarity see Section 2.4.1.

An early attempt to overcome the untenable orthogonality assumption of the vector space model was proposed under the name of generalized vector space model (Wong et al., 1985). The article which proposed the model did not provide empirical results, and since then the model has been regarded of large theoretical importance with less impact on actual applications. The model takes a distri-

butional approach, focusing on term co-occurrences. The underlying assumption is that term correlations are captured by the co-occurrence information. That is, two terms are semantically related if they co-occur often in the same documents. By eliminating orthogonality, documents can be seen as similar even if they do not share any terms. The term co-occurrence matrix is AA', hence the model takes A' as the semantic similarity matrix S. A major drawback of the generalized vector space model is that it replaces the orthogonality assumption with another questionable assumption. The computational needs are tremendous too, if the dimensions of A are considered. Moreover, the co-occurrence matrix is not sparse anymore.

Latent semantic indexing (or latent semantic analysis) was another attempt to bring more linguistic and psychological aspects to language processing via a kernel. Conceptually, latent semantic indexing is similar to the generalized vector space model, it measures semantic information through co-occurrence analysis in the corpus. From the algorithmic perspective it is an enormous problem that textual data have a large number of relevant features. This results in huge computational needs and the classification models may overfit the data. The number of features can be reduced by multivariate feature extraction methods. In latent semantic indexing, the dimension of the vector space is reduced by singular value decomposition (Deerwester et al., 1990).

Using rank reduction, terms that occur together very often in the same documents are merged into a single dimension of the feature space. The dimensions of the reduced space correspond to the axes of greatest variance. For latent semantic indexing, by dual representation the kernel matrix is $K = V\Sigma_k^2 V'$, where Σ_k is a diagonal matrix containing the k largest singular values of the singular value decomposition of the vector space, and V holds the right singular vectors of the decomposition. The new kernel matrix can be obtained directly from K by applying an eigenvalue decomposition of K (Cristianini et al., 2002). The computational complexity of performing an eigenvalue decomposition on the kernel matrix is a major drawback of latent semantic indexing.

2.3 Text Representation Enrichment Strategies by Term Expansion

In order to eliminate the bottleneck of the traditional BOW representation, previous approaches in term expansion enriched this convention by external lexical resources such as WordNet.

As a first step, these methods generate new features for each document in the dataset. These new features can be synonyms or homonyms of document terms as in (Hotho et al., 2003; Rodriguez and Hidalgo, 1997), or expanded features for terms, sentences and documents as in (Gabrilovich and Markovitch, 2005), or term context information for word sense disambiguation such as topic signatures (Agirre and De Lacalle, 2003; Agirre et al., 2004).

Then, the generated new features replace the old ones or are appended to the document representation, and construct a new vector representation $\hat{\mathbf{a}}_i$ for each text document. The similarity measure of document pairs is defined as:

$$S(\hat{\mathbf{a}}_i, \hat{\mathbf{a}_j}) = \frac{\hat{\mathbf{a}}_i \hat{\mathbf{a}_j}}{|\hat{\mathbf{a}}_i||\hat{\mathbf{a}_j}|}. \quad (2)$$

2.4 Our Framework

The basic assumption of our framework is that terms can be arranged in an order such that consecutive terms are semantically related. Hence each term acquires a unique position, and this position ties the term to its semantically related neighbors. However, given a BOW representation with a cosine similarity measure, this position would not improve classification performance. Therefore we suggest to associate a mathematical function with each term, thus mapping terms and documents to the L_2 space, and using the inner product of this space to express similarity. The choice of function will determine to which extent neighboring terms, i.e., the enriching terms, are considered in calculating the similarity between two documents. This section first introduces an algorithm that produces the aforementioned semantic order, then the semantic kernels in the L_2 space are discussed.

2.4.1 An Algorithm for a Semantic Ordering of Terms

The proposed kernels assume that there is a semantic order between terms. Let V denote a set of

terms $\{t_1, t_2, \ldots, t_n\}$ and let $d(t_i, t_j)$ denote the semantic distance between the terms t_i and t_j. The initial order of the terms is not relevant, though it is assumed to be alphabetic. Let $G = (V, E)$ denote a weighted undirected graph, where the weights in the set E are defined by the distances between the terms.

Various lexical resource-based (Budanitsky and Hirst, 2006) and distributional measures (Mohammad and Hirst, 2005) have been proposed to measure semantic relatedness and distance between terms. Terms can be corpus- or genre-specific. Manually constructed general-purpose lexical resources include many usages that are infrequent in a particular corpus or genre of documents. For example, one of the 8 senses of *company* in WordNet is a *visitor/visitant*, which is a hyponym of *person* (Lin, 1998). This sense of the term is practically never used in newspaper articles, hence distributional attributes should be taken into consideration. Composite measures that combine the advantages of both approaches have also been developed (Resnik, 1995; Jiang and Conrath, 1997). This paper relies on the Jiang-Conrath composite measure (Jiang and Conrath, 1997), which has been shown to be superior to other measures (Budanitsky and Hirst, 2006), and we also found that this measure works the best for the purpose. The Jiang-Conrath metric measures the distance between two senses by using the hierarchy of WordNet. By denoting the lowest superordinate of two senses s_1 and s_2 in the hierarchy with LSuper(s_1,s_2), the metric is calculated as follows:

$$d(s_1, s_2) = \mathrm{IC}(s_1) + \mathrm{IC}(s_2) - 2\mathrm{IC}(\mathrm{LSuper}(s_1, s_2)),$$

where IC(s) is the information content of a sense s based on a corpus. Distance between two terms is calculated according to the following equation: $\mathrm{d}(t_1, t_2) = \max_{s_1 \in \mathrm{sen}(t_1), s_2 \in \mathrm{sen}(t_2)} d(s_1, s_2)$, where t_1 and t_2 are two terms, and $\mathrm{sen}(t_i)$ is the set of senses of t_i. The distance between two terms is usually defined as the minimum of the sense distances. We chose maximum because it ensures that only closely related terms will be placed to adjacent positions by the algorithm below.

Finding a semantic ordering of terms can be translated to a graph problem: a minimum-weight Hamiltonian path G' of G gives the ordering by reading the nodes from one of the paths to the other. G is a complete graph, therefore such a path always exists, but finding it is an NP-complete problem. The following greedy algorithm is similar to the nearest neighbor heuristic for the solution of the traveling salesman problem. It creates a graph $G' = (V', E')$, where $V' = V$ and $E' \subset E$. This G' graph is a spanning tree of G in which the maximum degree of a node is two, that is, the minimum spanning tree is a path between two nodes.

Step 1 Find the term at the highest stage of the hierarchy in a lexical resource.

$$t_s = \mathrm{argmin}_{t_i \in V} \mathrm{depth}(t_i).$$

This seed term is the first element of V', $V' = \{t_s\}$. Remove it from the set V:

$$V := V \backslash \{t_s\}.$$

Using WordNet, this seed term is *entity*, if the vocabulary of the text collection contains it.

Step 2 Let t_l denote the leftmost term of the ordering and t_r the rightmost one. Find the next two elements of the ordering:

$$t'_l = \mathrm{argmin}_{t_i \in V} d(t_i, t_l),$$

$$t'_r = \mathrm{argmin}_{t_i \in V \backslash \{t'_l\}} d(t_i, t_r).$$

Step 3 If $d(t_l, t'_l) < d(t_r, t'_r)$ then add t'_l to V', $E' := E' \cup \{e(t_l, t'_l)\}$, and $V := V \backslash \{t'_l\}$. Else add t'_r to V', $E' := E' \cup \{e(t_r, t'_r)\}$ and $V := V \backslash \{t'_r\}$.

Step 4 Repeat from *Step 2* until $V = \emptyset$.

The above algorithm can be thought of as a modified Prim's algorithm, but it does not find the optimal minimum-weight spanning tree.

2.4.2 Semantic Kernels in the L_2 Space

The L_2 space shares resemblance with a real vector space. Real-valued vectors are replaced by square-integrable functions, and the dot product is replaced by the following inner product: $(f_i, f_j) = \int f_i f_j dx$, for some f_i, f_j in the given L_2 space.

Lately, Hoenkamp has also pointed out that the L_2 space can be used for information retrieval when

he introduced a Haar basis for the document space (Hoenkamp, 2003). He utilized a signal processing framework within the context of latent semantic indexing. In order to apply an L_2 representation for text classification, the problem is approached from a different angle than by Hoenkamp, taking discounting expansion terms as our point of departure.

Assigning a function $w(x-k)$ to the term in the kth position in a semantic order, a document j can be expressed as follows:

$$f_j(x) = \sum_{k=1}^{M} a_{kj} w(x-k), \tag{3}$$

where x is in $[1, M]$, and it is the variable of integration in calculating the inner product of the L_2; x can be regarded as a "dummy" variable carrying no meaning in itself. The above formula will be referred to as a document function. In the experiments, the function $\exp(-bx^2)$ was used as $w(x)$, with b as a free parameter reflecting the width of the function expressing how many neighboring expansion terms are considered.

The inner product of the $L_2[1, M]$ space is applied to express similarity between two documents in similar vein as the dot product does in a real-valued vector space:

$$(f_i, f_j) = \int_{[1,M]} f_i(x) f_j(x) dx, \tag{4}$$

where f_i and f_j are the representations of the documents in the L_2 space ($f_i, f_j \in L_2([1, M])$).

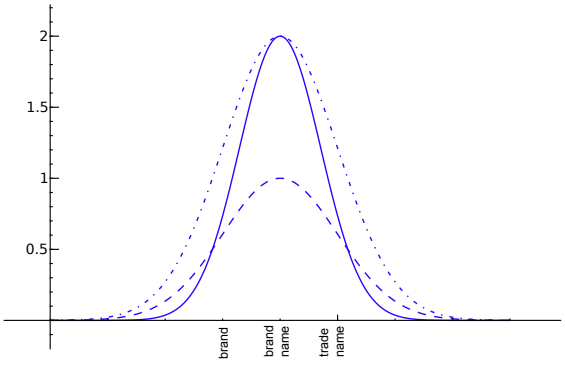


Figure 1: Two documents with matching term *brand name*. Dotted line: Document-1. Dashed line: Document-2. Solid line: Their product.

With the above formula, a matching term in two documents will be counted to its full term frequency or tfidf score, while semantically related terms will be counted less and less according their semantic similarity to the matching term. Assuming that the terms *brand*, *brand name*, and *trade name* follow each other in the semantic order, consider the following example. The first document has the term *brand name*, and so does the second document. In Figure 1, it can be seen brand name is counted the same way as it would be in a BOW model with its full term frequency score, brand and trade name are counted to a lesser extent, while other related terms are considered even less.

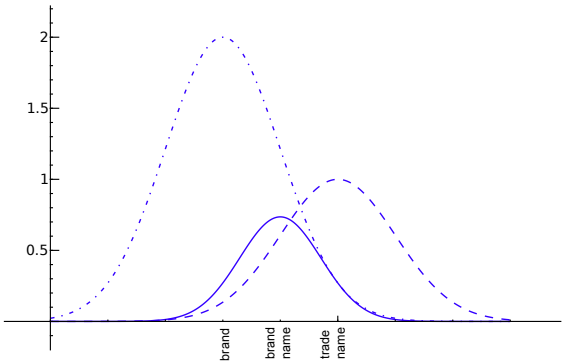


Figure 2: Two documents with no matching term but with related terms *brand* and *trade name*. Dotted line: Document-1. Dashed line: Document-2. Solid line: Their product.

Now if the two documents do not share the exact term, only related terms occur, for instance, *trade name* and *brand*, respectively, then the term *brand name*, placed between *trade name* and *brand* in the s semantic order, will be considered only to some extent for the calculation of similarity (see Figure 2).

3 Experimental Results

The most widely used benchmark corpus is the Reuters-21578 collection. For benchmarking purposes, the ModApte split was adopted. 9603 documents were used as the training set and 3299 as the test set in the experiments. Only those ninety text categories which had at least one positive example in the training set were included in the benchmark. Another benchmark data corpus we used was the 20

Newsgroups corpus, which is a collection of approximate 20,000 newsgroup documents nearly evenly divided among 20 discussion groups and each document is labeled as one of the 20 categories corresponding to the name of the newsgroup that the document was posted to.

In preparing the index terms, we restricted the vocabulary to the terms of WordNet 3.0 in order to be able to calculate the similarity score between any two terms. Stop words were removed in advance. Multiple word expressions were used to fully utilize WordNet. We used the built-in stemmer of WordNet, which is able to distinguish between different parts-of-speeches if the form of the word is unambiguous. For example, {accommodates, accommodated, accommodation} was stemmed to {accommodate, accommodate, accommodation}. We used term frequency as term weighting.

Prior to the semantic ordering, terms were assumed to be in alphabetic order. Measuring the Jiang-Conrath distance between adjacent terms, the average distance was 1.68 on the Reuters corpus. Note that the Jiang-Conrath distance was normalized to the interval $[0, 2]$. There were few terms with zero or little distance between them. This is due to terms which are related and start with the same word or stem. For example, *account, account executive, account for, accountable*.

The same average distance after reordering the terms with the proposed algorithm and the Jiang-Conrath distance was 0.56 on the same corpus. About one third of the terms had very little distance between each other. Nevertheless, over 10 % of the total terms still had the maximum distance. This is due to the non-optimal nature of the proposed term-ordering algorithm. These terms add noise to the classification. The noisy terms occur typically at the two sides of the scale, that is, the leftmost terms and the rightmost terms. While it is easy to find close terms in the beginning, as the algorithm proceeds, fewer terms remain in the pool to be chosen. For instance, *brand, brand name, trade name, label* are in the 33rd, 34th, 35th and 36th position on the left side counting from the seed respectively, while *windy, widespread, willingly, whatsoever, worried, worthwhile* close the left side, apparently sharing little in common. The noise can be reduced by the appropriate choice of the parameter b in $\exp(-bx^2)$, so that the impact of adjacent but distantly related terms can be minimized.

Kernel	Reuters Micro	Reuters Macro	20News Micro	20News Macro
Linear	0.900	0.826	0.801	0.791
Poly	0.903	0.824	0.796	0.788
L_2	0.911	0.835	0.813	0.799

Table 1: Micro- and macro-average F_1 results

Table 1 shows the results on the two benchmark corpora with the baseline linear kernel. Precision and recall with regard to a class c_k, the F_1 score shown is their average. For all the kernels, the results with the best parameter settings are shown. Polynomial kernels were benchmarked between degrees 2 and 5. L_2 kernels were benchmarked with width b between 1 and 8, the performance peaking at 4 in all cases. The model is able to outperform the baseline kernels, and the differences in micro-averaged results are statistically significant. In all cases of the L_2 kernel, the increase of F_1 was due the increase in both precision and recall.

4 Conclusions

Information systems are in great need of automated intelligent tools, but existing algorithms and methods cannot be pushed much further. Most techniques in current use are impaired by the semantically poor but widespread representation of information and knowledge. For this reason, we propose a new formalism that combines Cruse's idea about a sense spectrum, approximated by semantic ordering, and its calculation by functions.

The suggested model combines term expansion with the semantic relations and semantic relatedness used in semantic smoothing kernels. This slightly unusual approach needs to transform the real vector representation to the L_2 space, and the experimental results show that this new representation can improve text classification effectiveness.

Our new model also blends insights from different approaches to lexical semantics theory at its different levels. First, during the semantic ordering of terms the distributional hypothesis meets handcrafted lexical resources of word meaning that relate to term occurrences as if they were their referents,

a component external to term context. While high-quality lexical resources enable such an ordering in themselves, the procedure can benefit from data derived from the specific corpora in study – semantic relatedness measures such as the Jiang-Conrath similarity operate this way. Secondly, once the ordering is done and a sense spectrum is constructed, weights expressing statistical relationships between terms and documents are borrowed from the vector space model to form the basis for constructing hypothetical signals of content, documents as continuous functions.

5 Future research

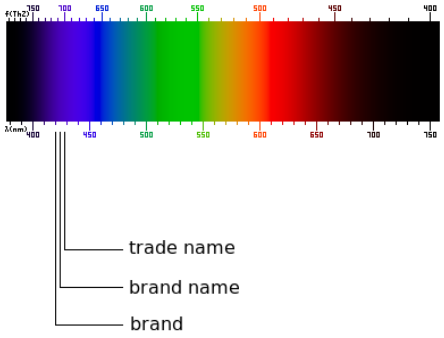


Figure 3: A hypothetical spectrum of terms.

As we have shown, a spectral interpretation of sense granularity can lead to improved text categorization results by utilizing L_2 space for information representation. Whether non-periodic functions other than the variant tested in this paper can be applied to the same end needs to be explored.

Turning back to the use of the spectrum of visible light for representing meaning, this raises new research questions. On the one hand, translating one-dimensional semantic ordering into colors is straightforward. Consider the following mapping. Assume that a language has a finite N number of terms, so the 1-dimensional result is an ordered list $o_1, o_2, \ldots, o_N$. Calculate the following: $\Delta = \sum_{i=1}^{N-1} d(o_i, o_{i+1})$, where Δ is the sum of distances between consecutive words. Further let $F : [0, \Delta] \rightarrow [400, 700]$ be the following mapping: $F(x) = 400 + x\frac{300}{\Delta}$. The visible spectrum is between 400 and 700 nm, F maps the cumulative distances of terms from $[0, \Delta]$ to the visible spectrum congruently, i.e. without distorting the distances. With this bijective (one-to-one) mapping, each term is assigned a physical wavelength and frequency. Figure 3 shows an example of such a term spectrum.

On the other hand, we have only begun to test the applicability of periodic functions in L_2 space (Wittek and Darányi, 2007), hence a well-established link to semantic computing by waves is missing for the time being. A possible compromise between the non-periodic vs. periodic approaches can be to apply wavelets instead of waves, a direction where our ongoing research shows promising results. These will be reported elsewhere. In a broader frame of thought, we are also working on the optical equivalents of the vector space model and the generalized vector space model as a first step toward coding more semantics in mathematical objects, and putting them to work in novel computing environments.

6 Acknowledgments

The authors are grateful to Martha Palmer (University of Colorado Boulder) for her inspiring suggestions and advice on sense granularity.

References

E. Agirre and O.L. De Lacalle. 2003. Clustering WordNet word senses. In *Proceedings of RANLP-03, 4th International Conference on Recent Advances in Natural Language Processing*, pages 121–130.

E. Agirre, E. Alfonseca, and O.L. de Lacalle. 2004. Approximating hierarchy-based similarity for WordNet nominal synsets using topic signatures. In *Proceedings of GWC-04, 2nd Global WordNet Conference*, pages 15–22.

R. Basili, M. Cammisa, and A. Moschitti. 2005. Effective use of WordNet semantics via kernel-based learning. In *Proceedings of CoNLL-05, 9th Conference on Computational Natural Language Learning*, pages 1–8.

S. Bloehdorn, R. Basili, M. Cammisa, and A. Moschitti. 2006. Semantic kernels for text classification based on topological measures of feature similarity. *Proceedings of ICDM-06, 6th IEEE International Conference on Data Mining*.

A. Budanitsky and G. Hirst. 2006. Evaluating WordNet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

N. Cristianini, J. Shawe-Taylor, and H. Lodhi. 2002. Latent semantic kernels. *Journal of Intelligent Information Systems*, 18(2):127–152.

D.A. Cruse. 1986. *Lexical semantics*.

S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

C. Dorrer, P. Londero, M. Anderson, S. Wallentowitz, and IA Walmsley. 2001. Computing with interference: all-optical single-query 50-element database search. In *Proceedings of QELS-01, Quantum Electronics and Laser Science Conference*, pages 149–150.

E. Gabrilovich and S. Markovitch. 2005. Feature generation for text categorization using world knowledge. In *Proceedings of IJCAI-05, 19th International Joint Conference on Artificial Intelligence*, volume 19.

E. Hoenkamp. 2003. Unitary operators on the document space. *Journal of the American Society for Information Science and Technology*, 54(4):314–320.

A. Hotho, S. Staab, and G. Stumme. 2003. WordNet improves text document clustering. In *Proceedings of SIGIR-03, 26th ACM International Conference on Research and Development in Information Retrieval*.

J. Hu, L. Fang, Y. Cao, H.J. Zeng, H. Li, Q. Yang, and Z. Chen. 2008. Enhancing text clustering by leveraging Wikipedia semantics. In *Proceedings of SIGIR-08, 31st ACM International Conference on Research and Development in Information Retrieval*, pages 179–186.

J.J. Jiang and D.W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*, pages 19–33.

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL Workshop on Usage of WordNet in Natural Language Processing Systems*, volume 98, pages 768–773.

D. Mavroeidis, G. Tsatsaronis, M. Vazirgiannis, M. Theobald, and G. Weikum. 2005. Word sense disambiguation for exploiting hierarchical thesauri in text classification. *Proceedings of PKDD-05, 9th European Conference on the Principles of Data Mining and Knowledge Discovery*, pages 181–192.

S. Mohammad and G. Hirst. 2005. Distributional measures as proxies for semantic relatedness.

H. Paijmans. 1997. Gravity wells of meaning: detecting information-rich passages in scientific texts. *Journal of Documentation*, 53(5):520–536.

M. Palmer, H.T. Dang, and C. Fellbaum. 2006. Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*, 13(02):137–163.

V.V. Raghavan and S.K.M. Wong. 1986. A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science*, 37(5):279–287.

P. Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of IJCAI-95, 14th International Joint Conference on Artificial Intelligence*, volume 1, pages 448–453.

J. Rodd, G. Gaskell, and W. Marslen-Wilson. 2002. Making sense of semantic ambiguity: Semantic competition in lexical access. *Journal of Memory and Language*, 46(2):245–266.

M.D.E.B. Rodriguez and J.M.G. Hidalgo. 1997. Using WordNet to complement training information in text categorisation. In *Procedings of RANLP-97, 2nd International Conference on Recent Advances in Natural Language Processing*.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*.

S. Shi, J.R. Wen, Q. Yu, R. Song, and W.Y. Ma. 2005. Gravitation-based model for information retrieval. In *Proceedings of SIGIR-05, 28th ACM International Conference on Research and Development in Information Retrieval*, pages 488–495. ACM New York, NY, USA.

G. Siolas and F. d'Alché Buc. 2000. Support vector machines based on a semantic kernel for text categorization. In *Proceedings of IJCNN-00, IEEE International Joint Conference on Neural Networks*.

C. J. van Rijsbergen. 2004. *The Geometry of Information Retrieval*.

P. Wittek and S. Darányi. 2007. Representing word semantics for IR by continuous functions. In S. Dominich and F. Kiss, editors, *Proceedings of ICTIR-07, 1st International Conference of the Theory of Information Retrieval*, pages 149–155.

P. Wittek, C.L. Tan, and S. Darányi. 2009. An ordering of terms based on semantic relatedness. In H. Bunt, editor, *Proceedings of IWCS-09, 8th International Conference on Computational Semantics*.

S.K.M. Wong, W. Ziarko, and P.C.N. Wong. 1985. Generalized vector space model in information retrieval. In *Proceedings of SIGIR-85, 8th ACM International Conference on Research and Development in Information Retrieval*, pages 18–25.

A simple feature-copying approach for long-distance dependencies

Marc Vilain, Jonathan Huggins, and Ben Wellner
The MITRE Corporation
202 Burlington Rd
Bedford, MA 01730 (USA)
{mbv,jhuggins,wellner}@mitre.org

Abstract

This paper is concerned with statistical methods for treating long-distance dependencies. We focus in particular on a case of substantial recent interest: that of long-distance dependency effects in entity extraction. We introduce a new approach to capturing these effects through a simple feature copying preprocess, and demonstrate substantial performance gains on several entity extraction tasks.

1 Long-distance dependencies

The linguistic phenomena known as long-distance dependencies have a long history in computational linguistics. Originally arising in phrase-structure grammar, the term aptly describes phenomena that are not strictly grammatical, and has thus gained currency in other endeavors, including that of concern to us here: entity extraction. The common thread, however, is simply that the treatment of a linguistic constituent α might be influenced by the treatment of a non-local constituent β.

In phrase-structure grammar, dependencies arise between matrix phrases and the gapped phrases that they dominate, as in "the cake that I hope you'll serve ε". The idea that these are *long-distance* dependencies arises from the fact that the separation between linked constituents can be arbitrarily increased while their dependency continues to hold (as in "the cake that I hope you'll ask Fred to tell Joan to beg Maryanne to serve ε").

With entity extraction, long-distance dependencies typically occur between mentions of the same entity. Consider, for example, the italicized references to Thomas White in this newswire excerpt:

> Bank of America on Friday named *Thomas White* head of global markets. *White* has been global head of credit products.

The fact that the first of these mentions is easily understood as person-denoting has substantial bearing on interpreting the second mention as person-denoting as well. But while local evidence for personhood is abundant for the first instance (*e.g.*, the given name "Thomas" or the verb "named"), the evidence local to the second instance is weak, and it is highly unlikely that a learning procedure would on its own acquire the relevant 5-gram context (α has been $\beta_{JJ}\ \gamma_{title}$). The dependency between these instances of *White* is thus a significant factor in interpreting both as names.

It is well known that capturing this kind of dependency can dramatically improve the performance of entity extraction systems. In this paper, we pursue a very simple method that enables statistical models to exploit these long-distance dependencies for entity extraction. The method obtains comparable or better results than those achieved by more elaborate techniques, and while we focus here on the specific case of entity extraction, we believe that the method is simple and reliable enough to apply generally to other long-distance phenomena.

2 Approaches to name dependencies

The problem of capturing long-distance dependencies between names has a traditional heuristic solution. This method, which goes back to systems participating in the original MUC-6 evaluation (Sundheim, 1995), is based on a found names list. The method requires two passes through the input. A first pass captures named entities based on local

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 192–200,
Boulder, Colorado, June 2009.

evidence, and enters these names into a found names registry. A second pass identifies candidate entities that were missed by the first pass, and compares them to entries in the registry. Where there is string overlap between the candidate and a previously found name, the entity type assigned to the existing entry is copied to the candidate.

Overall, this is an effective strategy, and we used it ourselves in a rule-based name tagger from the MUC-6 era (Vilain and Day, 1996). The strategy's Achilles heel, however, is what happens when erroneous entries are added to the found names list. These can get copied willy-nilly, thereby drastically increasing the scope of what may originally have started as a single local error. Clearly, the approach is begging to be given a firmer evidence-weighing foundation.

2.1 A statistical hybrid

An early such attempt at reformulating the approach is due to Minkheev *et al* (1999). As with previous approaches, Mikheev and his colleagues use a rule-based first pass to populate a found-names list. The second pass, however, is based on a maximum entropy classifier that labels non-first-passed candidates based on evidence accrued from matching entries on the found-names list. The statistical nature of the decision eliminates some of the failure modes of the heuristic found-names strategy, and in particular, prevents the copying of single errors committed in the first pass. The major weakness of the approach, however, is the heuristic first pass. Minkheev *et al* note that their method is most effective with a high-precision found-names list, implemented as a tightly controlled (but incomplete) rule-based first pass.

2.2 Fully-statistical models

Several more recent efforts have attempted to remove the need for a heuristic first-pass tagger, and have thus cast the problem as one-pass statistical models (Bunescu and Mooney, 2004; Sutton and McCallum, 2004; Finkel *et al*, 2005). While the technical details differ, all three methods approach the problem through conditional random fields (CRFs). In order to capture the long-distance dependencies between name instances, these approaches extend the linear-chain sequence models that are typically used for extracting entities with a CRF (Sha and Pereira, 2003). The resulting models consist of sentence-length sequences interlinked on those words that might potentially have long-distance interactions. Because of the graph-like nature of these models, the simplifying assumptions of linear-chain CRFs no longer hold. Since complete parameter estimation is intractable under these conditions, these three approaches introduce approximate methods for parameter estimation or decoding (Perceptron training for the first, loopy belief propagation for the first two, Gibbs sampling and simulated annealing for the third).

Krishnan and Manning (2006) provide a lucid critique of these extended models and of their computational ramifications. In a nutshell, their critique centers on the complexity of constructing the linked graphs (which they deemed high), the stability of Perceptron training (potentially unstable), and the run-time cost of simulated annealing (undesirably high). Since these undesirable properties are directly due to the treatment of long-distance dependencies through graphical models, it is natural to ask whether graphical models are actually required to capture these dependencies.

2.3 Avoiding non-sequential dependencies

In point of fact, Krishnan and Manning (2006) present an alternative to these graph-based methods. In particular, they break the explicit links that mutually condition non-adjacent lexemes, and instead rely on separate passes in a way that is reminiscent of earlier methods. A first-pass CRF is used to identify entities based solely on local information. The entity labels assigned by this first CRF are summarized in terms of lexeme-by-lexeme majority counts; these counts are then passed to a second CRF in the form of lexical features.

Consider, for example, a financial news source, where we would expect that a term like "Bank" might be assigned a preponderance of ORG labels by the first-pass CRF. This would be signaled to the second-pass CRF through a *token majority* feature that would take on the value *ORG* for all instances of the lexeme "Bank". This effectively aggregates local first-pass labeling decisions that apply to this lexeme, and makes the second-pass CRF sensitive to these first-pass decisions. Further refinements capture cases where a lexeme's label diverges from the token majority, for example: "Left Bank," where "Bank" will be assigned a LOC-valued *entity majority* feature whenever it ap-

pears in that particular word sequence. By capturing long-distance dependencies through lexical features, Krishnan and Manning avoid the need for graphical models, thus regaining tractability.

How well does this work? Returning to our earlier example, the idea behind these majority count features is that a term like "White" might be assigned the PER label by the first CRF when it appears in the context "Thomas White." Say, for the sake of argument, that sufficiently many instances of "White" are labeled PER by the first pass to sum to a majority. The second-stage CRF might then be expected to exploit the majority count features for "White" to PER-label any instances of White that were left unlabeled in the first pass (or that were given erroneous first-pass labels).

The method would be expected to fail, however, in cases where the first pass yields a majority of erroneous labels. Krishnan and Manning suggest that this is a fairly unlikely scenario, and demonstrate that their approach effectively captures long-distance name dependencies for the CoNLL English name-tagging task. They measured a best-in-class error reduction of 13.3% between their two-pass method and a single-stage CRF equipped with comparable features.

3 A contradictory data set

Just how unlikely, however, is the majority-error scenario that Krishnan and Manning discount? As it turns out, we encountered precisely this scenario while working with a corpus that is closely related to the CoNLL data used by Krishnan and Manning.

The corpus in question was drawn from the online edition of Reuters business news. The articles cover a range of business topics: mergers and acquisitions (M+A), stock valuations, management change, and so forth. This corpus is highly pertinent to this discussion, as the CoNLL English data are also Reuters news stories, drawn from the general news distribution. Our business data thus represent a natural branch of the overall CoNLL data.

A characteristic of these Reuters business stories that distinguishes them from general news is the prevalence of organization names, in particular company names. In these data, instances of company names significantly outnumber the next-most-common entities (money, dates, and the like). Even state-of-the-art CRFs trained on these data therefore err on the side of generating companies, meaning that in the absence of countermanding evidence (such as the presence of a person's given name), an entity will tend to be labeled ORG by default. Our earlier "Thomas White" example is a case in point: where the full name would typically be labeled PER, last-name-only instances ("White") might go unlabeled or be marked ORGs.

Label accuracy	count	% test cases
Trivially correct (present in both test and training)	6	—
Majority correct, test only	13	45%
Majority incorrect, test only	11	38%
Equivocal, test only	5	17%

Table 1: effectiveness of majority counts as predictors of entity type, Reuters business news sample

Table 1, above, shows a qualitative analysis of this phenomenon for PER entities in our M+A test set. The table considers person-denoting entities with three or more instances in the test set (n=35), and summarizes the majority accuracy of the labels assigned to them by a feature-rich 1-pass CRF. Of these thirty-five cases, we eliminate from consideration six trivial test cases that are present unambiguously in the training data (*e.g.*, "Carl Icahn"), since the CRF will effectively memorizes these cases during training. Of the remaining twenty-nine non-trivial cases, not quite half of them (45%) were accurately labeled by the CRF for the majority of their instances. A larger number of entities either received an incorrect majority label (38%) or were equivocally labeled, receiving an equal number of correct and incorrect tags (17%).

For this data set then, majority count features are poor models of the long-distance dependencies between person names, as they are just about as likely to predict the wrong label as the correct one.

4 A feature-copying alternative

A further analysis of our business news test sample revealed an intriguing fact. While in the absence of compelling evidence, the CRF might label a mention of a person entity as an org (or leave it unlabeled), for those mentions where compelling evidence existed, the CRF generally got it right. By compelling evidence, we mean such linguistic cues as the presence of a given name, contextual proximity to agentive verbs (*e.g.* "said"), and so forth.

This suggests an alternative approach to capturing these kinds of long-distance dependencies be-

tween names. In contrast to previous approaches, what is needed is not so much a way of coordinating non-local *decisions* about an entity's label, as a way of coordinating non-local *evidence* pertinent to the labeling decision. That is, instead of conditioning the labeling decision of a lexeme on the labeling decisions for that lexeme elsewhere in the corpus, we ought to condition the decision on the key evidence supporting those decisions.

4.1 Displaced features

Our approach operates by identifying those features of a CRF that are most predictive over a corpus. Each of those features is then duplicated: for a given token α, one version of the feature applies directly to α, while the other version applies to all other instances where α's word form appears in the current document. In particular, what we duplicate is the indicator function for a feature. The local version of an indicator Φ signals true if it applies locally to α, while the displaced version Φ_d signals true if it applies to *any* token α' that is an instance of the same word from as α.

To make this concrete, consider our opening example, now indexed with word positions:

Thomas$_7$ *White*$_8$ … *White*$_{13}$ has$_{14}$ been$_{15}$ …

Say that Φ is a feature indicator that is true of a token α_i just in case the token to its left, α_{i-1}, is a given name. In this instance, Φ(*White*$_8$) is true and Φ(*White*$_{13}$) is false. Then Φ_d, the displaced version of Φ, will be true of α_i just in case there is some token α_j with the same word form such that Φ(α_j) is true. In this instance Φ_d(*White*$_8$) and Φ_d(*White*$_{13}$) are both true by virtue of Φ being true of *White*$_8$.

This feature displacement scheme introduces non-local evidence into labeling decisions, effectively capturing the long-distance dependencies exhibited by name-tagging tasks. The method differs from previous approaches in that the models are not made conditional on non-local decisions (as in the case of graphical models), nor are they made conditional on aggregated first-pass decisions (as in Krishnan & Manning), but rather are made conditional on non-local evidence (displaced features).

4.2 Identifying features to displace

Because a typical entity extraction model can use tens or hundreds of thousands of features, it is not practical to displace every one of them. Though technically this only doubles the number of features under consideration, the lexical indexing rapidly gets out of hand. In addition, training and run times increase and, in our experience, a risk of over-fitting emerges. In point of fact, however, capturing long-distance name dependencies does not require us to replicate every last bit of feature-borne evidence. Instead, we only need to displace the evidence that is most reliably predictive.

To select predictive features to displace, we've had most success with a method based on information gain. Specifically, we use a one-time pre-process that measures feature gain relative to a corpus. The pre-process considers the same complement of feature schemas as are used by the actual CRF, and grounds the schemas on a training corpus to instantiate free lexical and P-O-S parameters. Gain for the instantiated features is measured through K-L divergence, and the *n* features with highest gain are then selected for displacement (with *n* typically ranging from 1,000 to 10,000).

As in (Schneider, 2004), gain for a given feature Φ, is found through a variant of the familiar Kullback-Leibler divergence formula,

$$D_{KL}(P \,\|\, Q) = \sum_i p(x_i) \log_2 \frac{p(x_i)}{q(x_i)}$$

For our purposes, the x_i are the non-null entity labels defined for the training set (PER, ORG, *etc.*), *P* is the probability distribution of the labels over the training set, *Q* is the distribution of the labels over tokens for which Φ applies, and *p* and *q* are their respective smoothed probability estimates (Laplace smoothing). Note in particular that this formulation excludes the null label ("not an entity"). This effectively means that K-L divergence is giving us a measure of the degree to which a feature predicts one or more non-null entity labels. Because the null label is generally the dominant label in named-entity tasks, including the null label in the calculation of K-L divergence tends to overwhelm the statistics, and leads to the selection of uninformative features that predict non-entities.

Figure 1 demonstrates the effectiveness of this feature selection method, along with sensitivity to the threshold parameter. The figure charts F-score on a Reuters business news task (M+A) as a function of the number of displaced features. From a baseline of F=89.3, performance improves rapidly with the addition of displaced features to the CRF model, reaching a maximum of F=91.4 with the

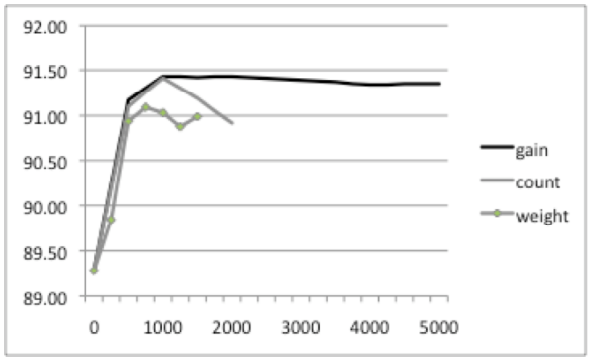


Figure 1: F score on the Reuters M+A task, as a function of number of displaced features

addition of 1,000 displaced features. Performance then fluctuates asymptotically around this level.

The chart also shows comparable growth curves for two alternative feature selection methods. The feature count method is similar to feature gain, but instead of ranking features with K-L divergence, it ranks them according to the number of times they match against the corpus. Feature weight does not use a schema-grounding first pass to generate candidate features, but trains a CRF model on the corpus, and then ranks features according to the weight assigned to them in the model. In preliminary experiments, neither of these methods yielded as high-performing a set of displaced features as feature gain. Additionally their growth curves exhibit sensitivity to parameter setting, which suggests a risk of over-fitting. For these reasons, we did not pursue these approaches further.

Note finally that the feature schemas we consider for displacement only encode local evidence (see Table 2 below). In particular, they do not encode the assigned label of a word form, as this would effectively introduce the kind of graphical conditional dependencies that lie outside the scope of linear-chain CRF methods.

4.3 Training and decoding

Aside from two pre-processing steps, training or decoding a CRF with displaced features is no different from training or decoding one with only conventional features. As to the pre-processing steps, the first applies to the corpus overall, as we must initially select a collection of locally predictive features to displace. The second step applies on a per-document basis and consists of the creation of the inverted lexical indices that are used to trigger indicator functions for displaced features.

While these additional steps complicate training and decoding somewhat, they have little effect on actual decoding run times. Most importantly, they retain the linear-chain properties of the CRF, and therefore do not require the graphical modeling and involved parameter estimation called for by most previous approaches. In addition, the training logistics are of a lesser magnitude than those required by Krishnan and Manning's approach, since training their second-stage model first requires round-robin training of one-fold-left-out classifiers that estimate first-stage majority counts.

5 Experimental design

To evaluate the effectiveness of feature copying with long-distance dependencies, we undertook a number of information extraction experiments. We focused on the traditional name-tagging task, relying on both current and archival data sets. For each data set, we trained entity-extraction models that corresponded to three different strategies for capturing long-distance dependencies.

- Baseline model: a feature-rich CRF trained with only local features and no long-distance dependency features;
- Feature-copying model: a CRF trained with the same local features, along with displaced versions of high-gain features;
- Majority model: a re-implementation of the Krishnan and Manning strategy, using the same feature set as the baseline CRF as well as their majority count features.

We used held-out development test sets to tune the selection of displaced features, in particular, the number of features to displace.

5.1 CRF configurations

We used the Carafe open-source implementation of sequence-based conditional random fields.[1] Carafe has achieved competitive results for standard sequence modeling tasks (Wellner & Vilain, 2006, Wellner *et al*, 2007), and allows for flexible feature design. Carafe provides several learning methods, including a fast gradient descent method using periodic step-size adjustment (Huang *et al*, 2007). Preliminary trials, however, produced better results

[1] http://sourceforge.net/projects/carafe

lexical unigrams	w_{-2} … w_{+2}
lexical bigrams	w_{-2},w_{-1} … w_{+1},w_{+2}
P-O-S unigrams	p_{-2} … p_{+2}
P-O-S bigrams	p_{-2},p_{-1} … p_{+1},p_{+2}
substrings	.*s or s.* $\|\|s\|\| \leq 4$
linguistic word lists	gazetteers, date atoms, …
regular expressions	caps., digits, …
"corp." nearby	also "ltd." …

Table 2: Baseline features; w_i and p_i respectively denote lexeme and P-O-S in relative position i.

Corpus	Language	NU	TM	MI	Topics
MUC-6	English	✓	✓		mostly politics
MUC-7	English	✓	✓r		mostly politics
MNET	Spanish	✓	✓r		mostly politics
Reuters	English	✓	✓		business
CoNLL	English			✓	all news

Table 3: Data set characteristics. All include persons, organizations, and locations; some have numeric forms (NU), dates and times (TM) where r indicates relative dates, or misc (MI).

with conditional log-likelihood learning (L-BFGS optimization). We used this latter method here, L2-regularized by a spherical Gaussian prior with variance set to 10.0 (based on preliminary trials).

Our baseline CRF was given a feature set that has proven its mettle in the literature (see Table 2). Along with contextual n-grams and the like, these features capture linguistic regularities through membership in vocabulary lists, *e.g.*, first names, major geographical names, honorifics, *etc.* They also include hand-engineered lists from our legacy rule-based tagger, *e.g.*, head word lists for organization names, lists of agentive verbs that reliably apply to persons, date atoms, and more. For part-of-speech features, we either accepted the parts of speech provided with a data set, or generated them with our implementation of Brill's method (Brill, 1994). For the majority count features, we used document and corpus versions the *token* and *entity* features described by Krishnan and Manning, but did not re-implement their *super-entity* feature.

5.2 Experimental data

We evaluated our approach on five different data sets: our current corpus of Web-harvested Reuters business news, as well as four archival data sets that have been reported on by other researchers. The business news data consist of a training corpus of mergers and acquisition stories (M+A), development and evaluation test sets for M+A and test sets for three additional topics: hot stocks (HS), new initiatives (NI), and general business news (BN). Table 3 provides an overview of our data sets and of some salient distinctions between them.

All five extraction tasks require the reporting of three core entity types: persons, organizations, and locations; additional required types are noted in the table. The reporting guidelines for the first four tasks are closely related: Reuters business and MUC-6 were annotated to the same original MUC-6 standard, while MUC-7 and MNET extend the MUC-6 standard slightly. The CoNLL standard alone calls for a catch-all (and troublesome) MISC entity.

5.3 Scoring metrics

Previous results on these data sets have been reported using one of two scoring methods: strict match (CoNLL) or match with partial credit, as calculated by the MUC scorer (MUC-6, MUC-7, and MNET). To enable comparisons to previously published work, we report our results with the metric appropriate to each data set (we use the MUC scorer for Reuters). These scoring distinctions are pertinent only to comparisons of absolute performance. In this paper, the interest is with relative comparisons across approaches to long-distance dependencies, for which the scorers are kept constant.

6 Experimental results

Table 4 summarizes our experimental results for the seven test sets annotated to the MUC-6 standard or its close variants (we will consider the CoNLL task separately). Along with F scores for our baseline CRF, the table presents F scores and baseline-relative error reduction (Δ_E) for two approaches to long-distance name dependencies: feature displacement (disp) and the Krishnan and Manning strategy (K+M). We were pleased to see that feature displacement proved effective for all of the extraction tasks. As the table shows, the addition of displaced features consistently reduced the residual error term left by the baseline CRF trained only with local features. For the English-language corpora, the error reduction ranged from a low of 11 % for the Reuters NI task to a high of 39% for the MUC-6 task. The error reduction for the Spanish-language MNET task was lowest of all, at 8.9%.

For all the English tasks, we consistently achieved better results with feature displacement

	MUC-6		MUC-7		MNET		Reuters M+A		Reuters BN		Reuters HS		Reuters NI	
	F	Δ_E	F	Δ_E	F	Δ_E	F	Δ_E	F	Δ_E	F	Δ_E	F	Δ_E
baseline	88.2	—	84.0	—	88.9	—	89.3	—	89.5	—	85.4	—	88.8	—
disp.	**92.8**	**39%**	**86.2**	**14%**	89.9	8.9%	**91.4**	**20%**	**91.8**	**22%**	**87.3**	**13%**	**90.1**	**11%**
K+M	91.5	28%	85.2	7.4%	—	—	90.4	11%	91.0	14%	86.3	6.2%	89.2	2.8%

Table 4: Performance on seven test sets annotated to variants of the MUC-6 standard (MUC scorer).

than with our version of Krishnan and Manning's approach (we were not able to obtain Spanish K+M results by publication time). In each case, displacement produced a greater reduction in baseline error than did majority counts. Furthermore, because both approaches start from the same baseline CRF, the resulting raw performance was consequently also higher for displacement. Note in particular the Reuters M+A test set: these are the data for which Table 1 suggests that majority counts would be poor predictors of long-distance effects. This prediction is in fact borne out by our results.

6.1 Effects of linguistic engineering

We were interested to note that the feature displacement method achieved both highest performance and highest error reduction for the MUC-6 corpus (F=92.8, Δ_E=39.3%) and for two of the Reuters test sets: M+A (F=91.4, Δ_E=20.0%) and BN (F=91.8, Δ_E=21.6%). The MUC-6 F-score, in particular, is comparable to those of hand-built MUC-era systems; in fact, it *exceeds* the score of our own hand-built MUC-6 system (Aberdeen *et al*, 1995).

What is apparently happening is that these three data sets are well matched to a group of linguistically inspired lexical features with which we trained our baseline CRF. In particular, our baseline features include gazetteers and word lists hand-selected for identifying entities based on local context: first names, agentive verbs, date atoms, *etc*. This played out in two significant ways. First, these linguistic features tended to elevate baseline performance (see Table 4). Second, these same features also proved effective when displaced, as demonstrated by the substantial error reduction with displacement. Feature displacement thus further rewards sound feature engineering.

6.2 Other MUC-related results

The MUC-7 and Reuters hot stocks data (HS) provide informative contrasts. For these data, feature displacement provided error reduction of Δ_E=13.9% and 13.4% respectively, which is less than for the top three data sets. It is interesting to note that in both cases, the baseline score is also lower, suggesting again that the performance of feature copying follows the performance of baseline tagging. In the case of Reuters HS, the evaluation data contained many out-of-training references to stock indices, which depressed baseline scores. Similar development-to-evaluation divergences have also been noted with the MUC-7 corpus.

6.3 The CoNLL task

Our results for the CoNLL task, reported in Table 5 below, provide a different point of contrast. The middle two rows of the table present the same experimental configurations as have been discussed so far. For this data set, we note that feature displacement does not perform as well as our reimplementation of Krishnan and Manning's strategy in terms of both absolute score and error reduction. Likewise, published results for other approaches mostly outperform displacement (see the first three rows in Table 5).

One possible explanation lies with the linguistic features with which we approached CoNLL: these are the same ones we originally developed for MUC-6. As noted earlier the CoNLL standard diverges in several ways from MUC-6. In particular, CoNLL calls for a MISC entity that covers a range of name-like entities, *e.g.*, events. MISC also, however, captures names that are trapped by tokenization ("London-based"), as well as some MUC organizations (sports leagues). This suggests that adapting our features to the CONLL task might help.

	base F	LDD F	Δ_E
Bunescu + Mooney 2004	80.09	82.30	11.1%
Finkel et al 2005	85.51	86.86	9.3%
Krishnan + Manning 2006	85.29	87.34	13.3%
K+M (re-impl, MUC feats.)	84.3	86.0	10.7%
displacement (MUC feats.)	84.3	85.8	9.6%
displ. (CoNLL feats.)	85.24	86.55	8.9%
displ. (CoNLL feats. + DS)	86.57	87.39	6.1%

Table 5: Performance on the CoNLL task; LDD designates use of long-distance dependency method.

The final two rows in Table 5 present attempts to tune our features to CoNLL. This includes some features (the "CoNLL feats" in Table 5) indicating story topic, all-caps headline contexts, presence in a sporting result table, and similar idiosyncrasies. In addition, we also used features based on distributional similarity word lists (DS in the table) provided with the Stanford NER package.[2]

While these feature engineering efforts proved effective, what we found surprised us. As Table 5 shows, the CoNLL features do substantially raise baseline performance, with the full set of new features producing a baseline (F=86.6) that outperforms previously published baselines by over a point of F score. In keeping with our observations for the MUC-annotated text, we would then have expected to see a comparable increase in the performance of displaced features, *i.e.*, a jump in error reduction relative to the baseline. Instead, we found just the reverse. Whereas displacement accounts for a 1.5 point gain in F (Δ_E=9.6%) with the MUC baseline features, with the beter CoNLL features, the gain due to displacement falls to 0.82 points of F (Δ_E=6.1%). While the final result with displacement (F=87.39) slightly edges out the previous high water mark of F=87.35 (Krishnan and Manning, 2005), the pattern is puzzling and not in keeping with our seven other data sets.

One possible explanations lies again with the CoNLL standard. The standard calls explicitly for inconsistent annotation of the same entity when used in different contexts. Along with place names being called MISC in hyphenated contexts (noted above), some places must be called ORG when used to refer to sports teams – except in results tables, where they are sometimes LOC. Such inconsistencies subvert the notion of long-distance dependencies by making these dependencies contradictory, thereby reducing the potential value of displacement as a means for improving performance.

7 Conclusions

Earlier in this paper, we introduced the notion of long-distance dependencies through their original codification in the context of phrase-structure grammars. By an interesting historical twist, the original solution to these grammatical long-distance effects, known as gap threading (Pereira, 1981), involved what is essentially a feature-copying operation, namely unification of constituent features. It is gratifying to note that the method presented here has illustrious predecessors.

Regarding the particular task of interest here, entity extraction, this paper conclusively shows that a simple feature-copying method provides an effective method for capturing long-distance dependencies between names. For the MUC-6 task, in particular, this error reduction is enough to lift a middle-of-the-pack performance from our baseline CRF to a level that would have placed it among the handful of top performers at the MUC-6 evaluation.

As noted, the method is also substantially more manageable than earlier approaches. It avoids the intractability of graphical models and also avoids the approximations required by methods that rely on these models. It also adds only minimal processing time at training and run times. This provides a practical alternative to the method of Krishnan and Manning, who require twelve separate training runs to create their models, and further require a time-consuming run-time process to mediate between their first and second stage CRFs.

We intend to take this work in two directions. First, we would like to get to the bottom of why the method did not do better with the CoNLL and MNET tasks. As noted earlier, our hypothesis is that we would expect greater exploitation of long-distance dependencies if we first improved the performance of the baseline CRF, especially by improving the acuity of task-related features. While it is not a key interest of ours to achieve best-in-class performance on historical evaluations, it is the case that we seek a better understanding of the range of application of the feature copying method.

Another direction of interest is to consider other problems that exhibit long-distance dependencies that might be addressed by feature copying. Word sense disambiguation is one such case, especially given Yarowsky's maxim regarding one sense per discourse, a consistency notion that seems tailor-made for treatment as long-distance dependencies (Yarowsky, 1995). Likewise, we are curious about the applicability of the method to reference resolution, another key task with long-distance effects.

Meanwhile, we believe that this method provides a practical approach for capturing long-distance effects in one of the most practical and useful application of human language technologies, entity extraction.

[2] http://nlp.stanford.edu/software/CRF-NER.shtml

References

John Aberdeen, John Burger, David Day, Lynette Hirschman, Patricia Robinson, and Marc Vilain. 1995. Description of the Alembic system as used for MUC-6. *Pcdgs of the 6th Message Understanding Conference* (MUC-6).

Eric Brill. 1994. Some advances in rule-based part-of-speech tagging. *Pcdgs. AAAI-94*.

Razvan Bunescu and Raymond J. Mooney. 2004. Collective information extraction with relational Markov networks. *Pcdgs. of the 42nd ACL*. Barcelona.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *Pcdgs. of the 43rd ACL*. Ann Arbor, MI.

Han-Shen Huang, Yu-Ming Chang, and Chun-Nan Hsu. 2007. Training conditional random fields by periodic step size adaptation for large-scale text mining. *Pcdgs. 7th Intl. Conf. on Data Mining (ICDM-2007).*

Vijay Krishnan and Christopher D. Manning. 2006. An Effective Two-Stage Model for Exploiting Non-Local Dependencies in Named Entity Recognition. *Pcdgs. of the 21st COLING and 44th ACL*. Sidney.

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. *Pcdgs. of the 9th EACL*. Bergen.

Fernando Pereira. 1981. Extraposition grammars. *American Jnl. of Computational Linguistics*, 4(7).

Karl-Michael Schneider. 2004. A new feature selection score for multinomial naive Bayes text classification based on KL-divergence. In *Companion to the Pcdgs. of the 42nd ACL.* Barcelona.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. *Pcdgs. of NAACL-HLT 2003*. Edmonton, CA.

Beth Sundheim, *ed.* 1995. *Pcdgs. of the 6th Message Understanding Conference (MUC-6)*. Columbia, MD.

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. *Pcdgs. ICML Workshop on Statistical Relational Learning*.

Marc Vilain and David Day. 1996. Finite-state phrse parsing by rule sequences. *Pcdgs. of the 16th Conference on Computational Lingusitics* (COLING-96).

Ben Wellner, Matt Huyck, Scott Mardis, John Aberdeen, Alex Morgan, Leon Peskin, Alex Yeh, Janet Hitzeman, and Lynette Hirschman. 2007. Rapidly retargetable approaches to de-identification. *Journal of the Americal Medical Informatics Association*; 14(5).

Ben Wellner and Marc Vilain. (2006). Leveraging machine-readable dictionaries in discriminative sequence models. In *Pcdgs. of the 5th Language Resources and Evaluation Conf.* (LREC 2006). Genoa.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. *Pcdgs. Of 33rd ACL.* Cambridge, MA.

Fine-Grained Classification of Named Entities Exploiting Latent Semantic Kernels

Claudio Giuliano
FBK-irst
I-38100, Trento, Italy
giuliano@fbk.eu

Abstract

We present a kernel-based approach for fine-grained classification of named entities. The only training data for our algorithm is a few manually annotated entities for each class. We defined kernel functions that implicitly map entities, represented by aggregating all contexts in which they occur, into a latent semantic space derived from Wikipedia. Our method achieves a significant improvement over the state of the art for the task of populating an ontology of people, although requiring considerably less training instances than previous approaches.

1 Introduction

Populating an ontology with relevant entities extracted from unstructured textual documents is a crucial step in Semantic Web and knowledge management systems. As the concepts in an ontology are generally arranged in deep class/subclass hierarchies, the problem of populating ontologies is typically solved top-down, firstly identifying and classifying entities in the most general concepts, and then refining the classification process.

Recent advances have made supervised approaches very successful in entity identification and classification. However, to achieve satisfactory performance, supervised systems must be supplied with a sufficiently large amount of training data, usually consisting of hand tagged texts. As domain specific ontologies generally contains hundreds of subcategories, such approaches are not directly applicable for a more fine-grained categorization because the number of documents required to find sufficient positive examples for all subclasses becomes too large, making the manual annotation very expensive.

Consequently, in the literature, supervised approaches are confined to classify entities into broad categories, such as persons, locations, and organizations, while the fine-grained classification has been approached with minimally supervised (e.g., Tanev and Magnini (2006) and Giuliano and Gliozzo (2008)) and unsupervised learning algorithms (e.g., Cimiano and Völker (2005) and Giuliano and Gliozzo (2007)).

Following this trend, we present a minimally supervised approach to fine-grained categorization of named entities previously recognized into coarse-grained categories, e.g., by a named-entity recognizer. The only training data for our algorithm is a few manually annotated entities for each class. For example, *Niels Bohr*, *Albert Einstein*, and *Enrico Fermi* might be used as examples for the class *physicists*. In some cases, training entities can be acquired (semi-) automatically from existing ontologies allowing us to automatically derive training entities for use with our machine learning algorithm. For instance, we may easily obtain tens of training entities for very specific classes, such as *astronomers*, *materials scientists*, *nuclear physicists*, by querying the Yago ontology (Suchanek et al., 2008).

We represent the entities using features extracted from the textual contexts in which they occur. Specifically, we use a search engine to collect such contexts from the Web. Throughout this paper, we will refer to such a representation as multi-context representation, in contrast to the single-context rep-

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 201–209,
Boulder, Colorado, June 2009.

resentation in which an entity is categorized using solely features extracted from the local context surrounding it, usually a window of a few words around the entity occurrence. Single-context features are commonly used in named-entity recognition, however to assign very specific categories the local context might not provide sufficient information. For example, in the sentence "Prof. Enrico Fermi discovered a way to induce artificial radiation in heavy elements by shooting neutrons into their atomic nuclei," single-context features such as, the prefix *Prof.* and the capital letters, provides enough evidence that Enrico Fermi is a person and a professor. However, to discover that he is a physicist we need to analyze a wider context, or alternatively multiple ones. Recently, Ganti et al. (2008) has shown that exploiting multi-context information can greatly improve the fine-grained classification of named entities, when compared to methods using single context only.

In order to effectively represent entities' multi-contexts, we extend the traditional vector space model (VSM), offering a way to integrate external semantic information in the classification process by means of latent semantic kernels (Shawe-Taylor and Cristianini, 2004). As a result, we obtain a generalized similarity function between multi-contexts that incorporates semantic relations between terms, automatically learned from unlabeled data. In particular, we use Wikipedia to build the latent semantic space. The underlying idea is that similar named entities tend to have a similar description in Wikipedia. As Wikipedia provides reliable information and it exceeds all other encyclopedias in coverage, it should be a valuable resource for the task of populating an ontology. To validate this hypothesis, we compare this model with one built from a news corpus.

Our approach achieves a significant improvement over the state of the art for the task of populating the People Ontology (Giuliano and Gliozzo, 2008), although requiring considerably less training instances than previous approaches. The task consists in classifying person names into a multi-level taxonomy composed of 21 categories derived from WordNet, making very fine-grained distinctions (e.g., physicists vs. mathematicians). It provides a more realistic and challenging benchmark than the ones previously available (e.g., Tanev and Magnini (2006) and Fleischman and Hovy (2002)), that consider a smaller number of categories arranged in a one-level taxonomy.

2 Entity Representation

The goal of our research is to determine the fine-grained categories of named entities requiring a minimal amount of human supervision.

Our method is based on the common assumption that named entities co-occurring with the same (domain-specific) terms are highly probable to refer to the same categories. For example, *quantum mechanics*, *atomic physics*, and *Nobel Prize in physics* are all terms that bound Niels Bohr and Enrico Fermi to the concept of physics.

To automatically derive features for the training and testing entities we proceed as follows. We pair each entity i with a multi-context m_i obtained by querying a search engine with the entity "i" and merging the first M snippets $s_{i,j}$ returned ($1 \leqslant j \leqslant M$). A multi-context is therefore a fictitious document obtained by aggregating snippets, i.e., summary texts of the search engine result. Formally, $m_i = \cup_{j=1}^{M} s_{i,j}$, where the operator $\cup$ denotes the concatenation of strings. For example, Figure 1 (a) and (b) show some snippets retrieved for "Enrico Fermi" and "Albert Einstein," while $s_1 \cup s_2 \cup s_3$ and $s_4 \cup s_5 \cup s_6$ represent their multi-contexts, respectively.

The following section describes how entities' multi-contexts are embedded into the feature space in order to train a kernel-based classifier.

3 Kernels for Fine-Grained Classification of Entities

The strategy adopted by kernel methods (Shawe-Taylor and Cristianini, 2004; Schölkopf and Smola, 2002) consists of splitting the learning problem in two parts. They first embed the input data in a suitable feature space, and then use a linear algorithm (e.g., the perceptron) to discover nonlinear pattern in the input space. Typically, the mapping is performed implicitly by a so-called *kernel function*. The kernel function is a similarity measure between the input data that depends exclusively on the specific data type and domain. A typical similarity function is the inner product between feature vectors. Characterizing the similarity of the inputs plays a crucial role in

s_1: [Enrico Fermi]$_{PER}$ discovered that many nuclear transformations could be conducted by using neutrons.
s_2: [Enrico Fermi]$_{PER}$ led the manhattan project's effort to create the first man-made and self-sustaining nuclear chain.
s_3: [Enrico Fermi]$_{PER}$ was most noted for his work on the development of the first nuclear reactor.

(a)

s_4: [Albert Einstein]$_{PER}$ did not directly participate in the invention of the atomic bomb.
s_5: [Albert Einstein]$_{PER}$ is one of the most recognized and well-known scientists of the century.
s_6: [Albert Einstein]$_{PER}$ was born at Ulm, in Württemberg, Germany, on March 14, 1879.

(b)

Figure 1: Examples of snippets retrieved for Enrico Fermi (a) and Albert Einstein (b).

determining the success or failure of the learning algorithm, and it is one of the central questions in the field of machine learning.

Formally, the kernel is a function $k : X \times X \rightarrow \mathbb{R}$ that takes as input two data objects (e.g., vectors, texts, parse trees) and outputs a real number characterizing their similarity, with the property that the function is symmetric and positive semi-definite. That is, for all $x_i, x_j \in X$, it satisfies

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \quad (1)$$

where ϕ is an explicit mapping from X to an (inner product) feature space $\mathcal{F}$.

In the next sections, we define and combine different kernel functions that calculate the pairwise similarly between multi-contexts. They are the only domain specific element of our classification system, while the learning algorithm is a general purpose component. Many classifiers can be used with kernels. The most popular ones are perceptron, support vector machines (SVM), and k-nearest neighbor (KNN).

3.1 Bag-of-Words Kernel

The simplest method to estimate the similarity between two multi-contexts is to compute the inner product of their vector representations in the VSM. Formally, we define a space of dimensionality N in which each dimension is associated with one word from the dictionary, and the multi-context m is represented by a row vector

$$\phi(m) = (f(t_1, m), f(t_2, m), \ldots, f(t_N, m)), \quad (2)$$

where the function $f(t_i, m)$ records whether a particular token t_i is used in m. Using this representation we define *bag-of-words kernel* between multi-contexts as

$$K_{BOW}(m_1, m_2) = \langle \phi(m_1), \phi(m_2) \rangle \quad (3)$$

However, the bag-of-words representation does not deal well with lexical variability. To significantly reduce the training set size, we need to map contexts containing semantically equivalent terms into similar feature vectors. To this aim, in the next section, we introduce the class of semantic kernels and show how to define an effective semantic VSM using (unlabeled) external knowledge.

3.2 Semantic Kernels

It has been shown that semantic information is fundamental for improving the accuracy and reducing the amount of training data in many natural language tasks, including fine-grained classification of named entities (Fleischman and Hovy, 2002), question classification (Li and Roth, 2005), text categorization (Giozzo and Strapparava, 2005), word sense disambiguation (Gliozzo et al., 2005).

In the context of kernel methods, semantic information can be integrated considering linear transformations of the type $\tilde{\phi}(c_j) = \phi(c_j)\mathbf{S}$, where $\mathbf{S}$ is a $N \times k$ matrix (Shawe-Taylor and Cristianini, 2004). The matrix $\mathbf{S}$ can be rewritten as $\mathbf{S} = \mathbf{WP}$, where $\mathbf{W}$ is a diagonal matrix determining the word weights, while $\mathbf{P}$ is the *word proximity matrix* capturing the semantic relations between words. The proximity matrix $\mathbf{P}$ can be defined by setting non-zero entries between those words whose semantic relation is inferred from an external source of domain knowledge. The *semantic kernel* takes the general form

$$\tilde{k}(m_i, m_j) = \phi(m_i)\mathbf{SS}'\phi(m_j)' = \tilde{\phi}(m_i)\tilde{\phi}(m_j)'. \quad (4)$$

It follows directly from the explicit construction that Equation 4 defines a valid kernel.

WordNet and manually constructed lists of semantically related words typically provide a simple way to introduce semantic information into the

kernel. To define a semantic kernel from such resources, we could explicitly construct the proximity matrix $\mathbf{P}$ by setting its entries to reflect the semantic proximity between the words i and j in the specific lexical resource. However, we prefer an approach that exploits unlabeled data to automatically build the proximity matrix, defining a language and domain independent approach.

3.2.1 Latent Semantic Kernel

To define a proximity matrix, we look at co-occurrence information in a (large) corpus. Two words are considered semantically related if they frequently co-occur in the same texts. We use singular valued decomposition (SVD) to automatically derive the proximity matrix $\mathbf{\Pi}$ from a corpus, represented by its term-by-document matrix $\mathbf{D}$, where the $\mathbf{D}_{i,j}$ entry gives the frequency of term t_i in document d_j.[1] SVD decomposes the term-by-document matrix $\mathbf{D}$ into three matrixes $\mathbf{D} = \mathbf{U\Sigma V}'$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices (i.e., $\mathbf{U}'\mathbf{U} = \mathbf{I}$ and $\mathbf{V}'\mathbf{V} = \mathbf{I}$) whose columns are the eigenvectors of $\mathbf{DD}'$ and $\mathbf{D}'\mathbf{D}$ respectively, and $\mathbf{\Sigma}$ is the diagonal matrix containing the singular values of $\mathbf{D}$.

Under this setting, we define the proximity matrix $\mathbf{\Pi}$ as follows:

$$\mathbf{\Pi} = \mathbf{U}_k\mathbf{\Sigma}_k, \tag{5}$$

where $\mathbf{U}_k$ is the matrix containing the first k columns of $\mathbf{U}$ and k is the dimensionality of the latent semantic space and can be fixed in advance. By using a small number of dimensions, we can define a very compact representation of the proximity matrix and, consequently, reduce the memory requirements while preserving most of the information.

The matrix $\mathbf{\Pi}$ is used to define a linear transformation $\pi : \mathbb{R}^N \rightarrow \mathbb{R}^k$, that maps the vector $\phi(m_j)$, represented in the standard VSM, into the vector $\tilde{\phi}(m_j)$ in the latent semantic space. Formally, π is defined as follows

$$\pi(\phi(m_j)) = \phi(m_j)(\mathbf{W\Pi}) = \tilde{\phi}(m_j), \tag{6}$$

where $\phi(m_j)$ is a row vector, $\mathbf{W}$ is a $N \times N$ diagonal matrix determining the word weights such that $\mathbf{W}_{i,i} = log(idf(w_i))$, where $idf(w_i)$ is the *inverse document frequency* of w_i.

[1] SVD has been first applied to perform latent semantic analysis of terms and latent semantic indexing of documents in large corpora by Deerwester et al. (1990).

Finally, the *latent semantic kernel* is explicitly defined as follows

$$K_{LS}(m_i, m_j) = \langle \pi(\phi(m_i)), \pi(\phi(m_j)) \rangle, \tag{7}$$

where ϕ is the mapping defined in Equation 2 and π is the linear transformation defined in Equation 6. Note that we have used a series of successive mappings each of which adds some further improvement to the multi-context representation.

3.3 Composite Kernel

Finally, to combine the two representations of multi-contexts, we define the composite kernel as follows

$$K_{BOW}(m_1, m_2) + K_{LS}(m_1, m_2). \tag{8}$$

It follows directly from the explicit construction of the feature space and from closure properties of kernels that it is a valid kernel.

4 Experiments

In this section, we compare performance of different kernel setups and previous approaches on an ontology population task.

4.1 Benchmark

Experiments were carried out on the People Ontology (Giuliano and Gliozzo, 2008). An ontology extracted from WordNet, containing 1,657 distinct person instances arranged in a multi-level taxonomy having 21 fine-grained categories (Figure 2). To provide a formal distinction between classes and instances, required to assign instances to classes, the authors followed the directives defined by Gangemi et al. (2003) for OntoWordNet, in which the informal WordNet semantics is re-engineered in terms of a description logic.

In order to have a fair comparison, we reproduced the same experimental settings used in Giuliano and Gliozzo (2008). The population task is cast as a categorization problem, trying to assign person instances to the most specific category. For each class, the instances were randomly split into two equally sized subsets. One is used for training and the other for test, and vice versa. The reported results are the average performance over these two subsets. When an instance is assigned to a sub-class it is also implicitly assigned to all its super-classes. For instance,

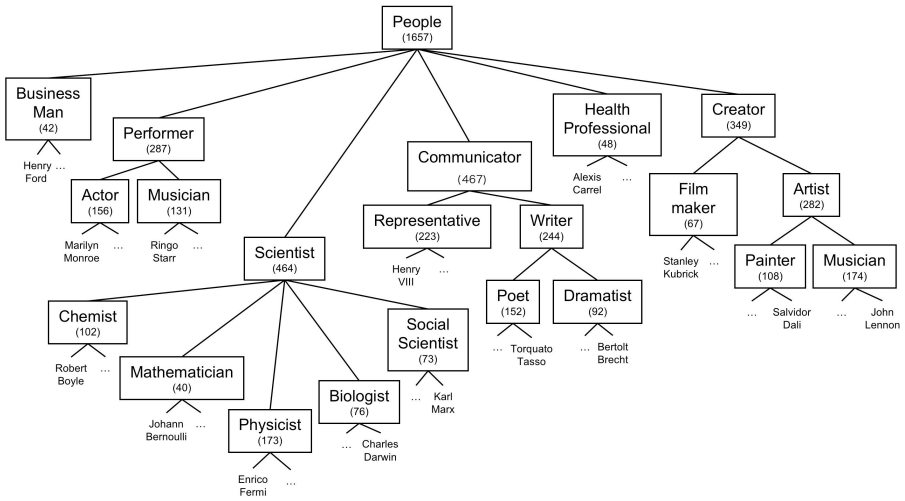


Figure 2: The People Ontology defined by Giuliano and Gliozzo (2008). Numbers in brackets are the total numbers of person instances per category. Concepts with less than 40 instances were removed.

classifying *Salvador Dali* as *painter* we implicitly classify him as *artist* and *creator*. The evaluation is performed as proposed by Melamed and Resnik (2000) for a similar hierarchical categorization task. For instance, classifying *John Lennon* as *painter*, we obtain a false positive for the spurious classification *painter*, a false negative for missing class *musician*, and two true positives for the correct assignment to the super-classes *artist* and *creator*.

4.2 Experimental Settings

We built two proximity matrices Π_W and Π_{NYT}. The former is derived from the 200,000 most visited Wikipedia articles, while the latter from 200,000 articles published by the New York Times between June 1, 1998 and January 01, 2000. After removing terms that occur less than 5 times, the resulting dictionaries contain about 300,000 and 150,000 terms respectively. We used the SVDLIBC package[2] to compute the SVD, truncated to 400 dimensions. To derive the multi-context representation, we collected 100 english snippets for each person instance by querying Google™. To classify each person instance into one of the fine-grained categories, we used a KNN classifier ($K = 1$). No parameter optimization was performed.

[2] `http://tedlab.mit.edu/~dr/svdlibc/`

4.3 Results

Table 1 shows micro- and macro-averaged results for K_{BOW}, K_W, $K_{BOW} + K_W$, K_{NYT}, $K_{BOW} + K_{NYT}$, the IBOP method (Giuliano and Gliozzo, 2008), the random baseline, and most frequent baseline.[3] Where K_W and K_{NYT} are instances of the latent semantic kernel, K_{LS}, using the proximity matrices Π_W and Π_{NYT}, derived from Wikipedia and the New York Times corpus, respectively. Table 2 shows detailed results for each sub- and super-category for $K_{BOW} + K_W$. Table 3 shows the confusion matrix of $K_{BOW} + K_W$, in which the rows are ground truth classes and the columns are predictions. The matrix has been calculated for the finer-grained categories and, then, grouped according to their super-class. To be compared with the IBOP method, all experiments were conducted using only 20 training examples per category. Finally, figure 3 shows the learning curves for $K_{BOW} + K_W$ obtained varying the number of snippets (12, 25, 50, and 100) used to derive the multi-contexts.

4.4 Discussion

On the one hand, the results (Table 2) show that learning the semantic model from Wikipedia gives no significant improvement. Therefore, we reject the hypothesis that encyclopedic knowledge can provide

[3] The most frequent category has been estimated on the training data.

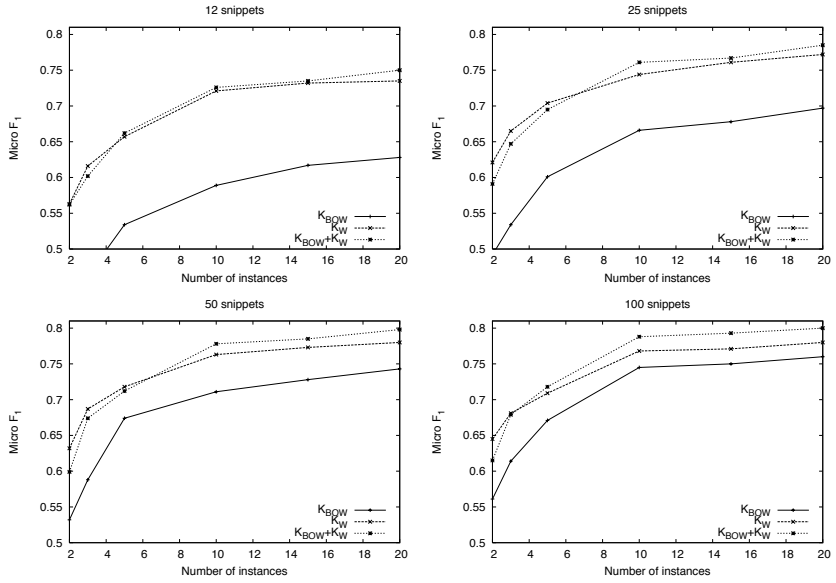


Figure 3: Learning curves for $K_{BOW} + K_W$ obtained varying the number of snippets used to derive the training and test sets. From top-left to bottom right: 12, 25, 50, and 100.

Method	Micro-$\mathbf{F_1}$	Macro-$\mathbf{F_1}$
K_{BOW}	75.6	70.6
K_W	78.1	73.1
$\mathbf{K_{BOW} + K_W}$	**80.0**	**75.4**
K_{NYT}	77.6	72.9
$K_{BOW} + K_{NYT}$	79.7	75.1
IBOP	70.1	62.3
Random	15.4	15.5
Most Frequent	20.7	3.3

Table 1: Comparison among the kernel-based approaches, the IBOP method (Giuliano and Gliozzo, 2008), the random baseline, and most frequent baseline.

more accurate semantic models than general purpose corpora. Moreover, further experiments have shown that even a larger number of Wikipedia articles (600,000) does not help. On the other hand, the latent semantic kernels outperform all the other methods, and their composite ($K_{BOW} + K_W$ and $K_{BOW} + K_{NYT}$) perform the best on every configuration, demonstrating the effectiveness of latent semantic kernels in fine-grained classification of named entities. As in text categorization and word sense disambiguation, they have proven effective tools to overcome the limitation of the VSM by introducing semantic similarity among words.

An important characteristic of the approach is the small number of training examples required per category. This affects both the prediction accuracy and the computation time (this is generally a common property of instance-based algorithms). The learning curves (Figure 3) show that the composite kernel ($K_{BOW} + K_{LS}$) obtained the same performance of the bag-of-word kernel (K_{BOW}) using less than half of the training examples per category. The difference is much more pronounced when using less snippets. The composite kernel $K_{BOW} + K_W$ reaches a plateau around 10 examples, and after 20 examples adding more examples does not significantly improve the classification performance.

As most of entities in the People Ontology are celebrities, all the snippets retrieved by Google™generally refer to them, alleviating the problem of ambiguity of proper names. However, person names are highly ambiguous. In a more realistic scenario, the result of a search engine for a person name is usually a mix of contexts about different entities sharing the same name. In this case, our approach have to be combined with a system that clusters the search engine result, where each cluster is assumed to contain all (and only those) contexts that refer to the same entity. The WePS evaluation campaign on disambiguation of person names (Artiles et al., 2007; Artiles et al., 2009) has shown that the best clustering systems achieve a precision of about 90%

	Scientist					Performer		Creator			Communicator			Business	Health
	Phy	Mat	Che	Bio	Soc	Act	Mus	Fil	Pai	Mus	Poe	Dra	Rep	man	prof
Phy	118	24	10	4	2	0	0	0	0	0	0	0	0	7	2
Mat	2	33	0	0	1	0	0	0	0	0	1	0	0	3	0
Che	13	2	68	9	2	0	0	0	0	0	0	0	0	5	2
Bio	3	0	7	52	0	0	0	0	1	0	0	0	1	6	6
Soc	0	4	1	1	55	0	0	0	0	0	3	1	1	4	2
Act	0	0	0	0	0	98	5	27	0	0	2	14	0	3	0
Mus	0	0	0	0	0	17	67	0	0	32	1	0	1	2	1
Fil	0	0	0	0	0	13	0	45	0	0	1	4	0	2	0
Pai	0	0	0	1	1	2	0	1	100	0	1	0	0	1	0
Mus	0	0	0	0	0	4	29	0	0	139	0	1	0	0	0
Poe	0	2	0	0	0	0	0	0	7	3	98	26	1	2	3
Dra	0	0	0	1	1	9	0	1	0	1	12	61	1	4	1
Rep	0	0	0	0	0	1	1	0	2	0	0	0	197	22	0
Bus	1	0	1	0	1	0	0	1	0	1	0	0	1	36	0
Hea	0	0	0	8	4	0	1	0	0	0	0	1	1	2	31

Table 3: Confusion matrix of $K_{BOW} + K_W$ for the more fine-grained categories grouped according to their top-level concepts of the People Ontology.

Category	**Prec.**	**Recall**	$\mathbf{F_1}$
Scientist	95.1	90.1	92.6
Physicist	86.1	70.7	77.6
Mathematician	50.8	82.5	62.9
Chemist	78.2	67.3	72.3
Biologist	68.4	68.4	68.4
Social scientist	82.1	76.4	79.1
Performer	75.7	69.3	72.3
Actor	68.1	65.8	66.9
Musician	65.0	55.4	59.8
Creator	78.9	82.6	80.7
Film Maker	60.0	69.2	64.3
Artist	83.6	85.4	84.5
Painter	90.9	93.5	92.2
Musician	79.0	80.3	79.7
Communicator	91.9	86.7	89.2
Representative	96.6	88.3	92.3
Writer	86.8	84.2	85.5
Poet	82.4	69.0	75.1
Dramatist	56.5	66.3	61.0
Business man	36.4	85.7	51.1
Health professional	64.6	64.6	64.6
micro	80.9	79.6	80.2
macro	75.1	76.3	75.7

Table 2: Results for each category using $K_{BOW} + K_W$.

and a recall of about 70% and that, in the majority of the cases, the number of contexts per entity is less than 20. This shows that latent semantic kernels are an effective tool for fine-grained classification of person names.

Finally, table 3 shows that misclassification errors are largely distributed among categories belonging to the same super-class (i.e., the blocks on the main diagonal are more densely populated than others). As expected, the algorithm is much more accurate for the top-level concepts (i.e., Scientist, Communicator, etc.), where the category distinctions are clearer, while a further fine-grained classification, in some cases, is even difficult for human annotators.

5 Related Work

Fleischman and Hovy (2002) approach the fine-grained classification of person instances using supervised learning, where the training set is generated semi-automatically, bootstrapping from a small training set. They compare different machine learning algorithms, providing local features as well as global semantic information derived from topic signature and WordNet. Person instances were classified into one of eight categories.

Cimiano and Völker (2005) present an approach for the fine-grained classification of entities relying on the Harris' distributional hypothesis and the vector space model. They assign a particular instance to the most similar concept representing both with lexical-syntactic features extracted from the context of the instance and the lexicalization of the concept, respectively. Experiments were performed using a large ontology with 682 concepts (unfortunately not yet available).

Tanev and Magnini (2006) proposed a weakly-supervised method that requires as training data a list of named entities, without context, for each category under consideration. Given a generic syntactically parsed corpus containing at least each training entity twice, the algorithm learns, for each category,

a feature vector describing the contexts where those entities occur. Then, it compares the new (unknown) entity with the so obtained feature vectors, assigning it to the most similar category. Experiments are performed on a benchmark of 5 sub-classes of location and 5 sub-classes of person.

Giuliano and Gliozzo (2007) propose an unsupervised approach based on lexical entailment, consisting in assigning an entity to the category whose lexicalization can be replaced with its occurrences in a corpus preserving the meaning. Using unsupervised learning, they obtained slightly worst results than Tanev and Magnini (2006) on the same benchmark.

Picca et al. (2007) present an approach for ontology learning from open domain text collections, based on the combination of Super Sense Tagging and Domain Modeling techniques. The system recognizes terms pertinent to the domain and assigns them the correct ontological type.

Giuliano and Gliozzo (2008) present an instance-based learning algorithm for fine-grained named entity classification based on syntactic features (word-order, case-marking, agreement, verb tenses, etc.). Their method can handle much finer distinctions than previous methods, and it is evaluated on a hierarchical taxonomy of 21 ancestors of people that was induced from WordNet. One contribution is to create this richer People Ontology. Another is to make effective use of the Web 1T 5-gram corpus (Brants and Franz, 2006) to represent syntactic information. The main difference between the two approaches lies primarily in the use of syntactic and semantic information. Our experiments show that semantic features do provide richer information than syntactic ones for a more fine-grained classification of named entities. In fact, the accuracy improvement achieved by our approach is more evident for the more specific classes. For example, the improvement in accuracy is about 14% for the class scientist, while it ranges from 25% to 46% for its sub-classes (physicist, mathematician, etc.).

Kozareva et al. (2008) propose an approach for person name categorization based on the domain distribution. They use the information provided by WordNet Domains to generated lists of words relevant for a given domain, by mapping and ranking the words from the WordNet glosses to their WordNet Domains. A named entity is then classified according the similarity between the word-domain lists and the global context in which the entity appears. However, the evaluation was performed only on 6 person names using two categories.

Ganti et al. (2008) present a method that considers an entity's context across multiple documents containing it, and exploiting word n-grams and existing large list of related entities as features. They generated training and test data using Wikipedia articles that contain list of instances. They compare their system with a single-context classifier, showing that their approach based on aggregate context perform better.

Finally, Talukdar et al. (2008) propose a graph-based semi-supervised label propagation algorithm for acquiring open-domain labeled classes and their instances from a combination of unstructured and structured text.

6 Conclusions

We presented an approach to automatic fine-grained categorization of named entities based on kernel methods. Entities are represented by aggregating all contexts in which they occur. We employed latent semantic kernels to extend the bag-of-words representation. The latent semantic models were derived from Wikipedia and a news corpus We evaluated our approach on the People Ontology, a multi-level ontology of people derived from WordNet. Although this benchmark is still far from being "large", it allows drawing more valid conclusions than past ones. We significantly outperformed the previous results on both coarse- and fine-grained classification, although requiring much less training instances. From this preliminary analysis, it appears that semantic information is much more effective that syntactic one for this task, and deriving the semantic model from Wikipedia gives no significant improvement, as well as, using a larger number of Wikipedia articles.

Acknowledgments

Claudio Giuliano is supported by the X-Media project (`http://www.x-media-project.org`), sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978 and the ITCH project (`http://itch.fbk.eu`), sponsored by the Italian Ministry of University and Research and by the Autonomous Province of Trento.

References

Javier Artiles, Julio Gonzalo, and Satoshi Sekine. 2007. The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 64–69, Prague, Czech Republic, June. Association for Computational Linguistics.

Javier Artiles, Julio Gonzalo, and Satoshi Sekine. 2009. Weps 2 evaluation campaign: overview of the web people search clustering task. In *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*, Madrid, Spain, April.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram corpus version 1, Linguistic Data Consortium, Philadelphia.

Philipp Cimiano and Johanna Völker. 2005. Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of RANLP'05*, pages 66– 166–172, Borovets, Bulgaria.

Scott C. Deerwester, Susan T. Dumais, Thoms 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.

Michael Fleischman and Eduard Hovy. 2002. Fine grained classification of named entities. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan.

Aldo Gangemi, Roberto Navigli, and Paola Velardi. 2003. Axiomatizing WordNet glosses in the OntoWordNet project. In *Proocedings of the Workshop on Human Language Technology for the Semantic Web and Web Services at ISWC 2003*, Sanibel Island, Florida.

Venkatesh Ganti, Arnd C. König, and Rares Vernica. 2008. Entity categorization over large document collections. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 274–282, New York, NY, USA. ACM.

Alfio Giozzo and Carlo Strapparava. 2005. Domain kernels for text categorization. In *Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 56–63, Ann Arbor, Michigan, June.

Claudio Giuliano and Alfio Gliozzo. 2007. Instance based lexical entailment for ontology population. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 248–256.

Claudio Giuliano and Alfio Gliozzo. 2008. Instance-based ontology population exploiting named-entity substitution. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 265–272, Manchester, UK, August.

Alfio Massimiliano Gliozzo, Claudio Giuliano, and Carlo Strapparava. 2005. Domain kernels for word sense disambiguation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 403–410, Ann Arbor, Michigan, June.

Zornitsa Kozareva, Sonia Vazquez, and Andres Montoyo. 2008. Domain information for fine-grained person name categorization. In *9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2008)*, pages 311–321, Haifa, Israel, 17-23 February.

Xin Li and Dan Roth. 2005. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249.

I. Dan Melamed and Philip Resnik. 2000. Tagger evaluation given hierarchical tag sets. *Computers and the Humanities*, pages 79–84.

Davide Picca, Alfio Gliozzo, and Massimiliano Ciaramita. 2007. Semantic domains and supersense tagging for domain-specific ontology learning. In David Evans, Sadaoki Furui, and Chantal Soulé-Dupuy, editors, *Recherche d'Information Assistée par Ordinateur (RIAO)*, Pittsburgh, PA, USA, May.

B. Schölkopf and A. Smola. 2002. *Learning with Kernels*. MIT Press, Cambridge, Massachusetts.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. YAGO: A large ontology from wikipedia and wordnet. *Elsevier Journal of Web Semantics*.

Partha Pratim Talukdar, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, Waikiki, Honolulu, Hawaii, October 25-27.

Hristo Tanev and Bernardo Magnini. 2006. Weakly supervised approaches for ontology population. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, Trento, Italy.

Using Encyclopedic Knowledge for Automatic Topic Identification

Kino Coursey
University of North Texas and Daxtron Laboratories, Inc.
kino@daxtron.com

Rada Mihalcea and William Moen
University of North Texas
rada,wemoen@unt.edu

Abstract

This paper presents a method for automatic topic identification using an encyclopedic graph derived from Wikipedia. The system is found to exceed the performance of previously proposed machine learning algorithms for topic identification, with an annotation consistency comparable to human annotations.

1 Introduction

With exponentially increasing amounts of text being generated, it is important to find methods that can annotate and organize documents in meaningful ways. In addition to the content of the document itself, other relevant information about a document such as related topics can often enable a faster and more effective search or classification. Document topics have been used for a long time by librarians to improve the retrieval of a document, and to provide background or associated information for browsing by human users. They can also assist search, background information gathering and contextualization tasks, and enhanced relevancy measures.

The goal of the work described in this paper is to automatically find topics that are relevant to an input document. We refer to this task as "topic identification" (Medelyan and Witten, 2008). For instance, starting with a document on "United States in the Cold War," we want to identify relevant topics such as "history," "Global Conflicts," "Soviet Union," and so forth. We propose an unsupervised method for topic identification, based on a biased graph centrality algorithm applied to a large knowledge graph built from Wikipedia.

The task of topic identification goes beyond keyword extraction (Mihalcea and Csomai, 2007), since relevant topics may not be necessarily mentioned in the document, and instead have to be obtained from some repositories of external knowledge. The task is also different from text classification (Gabrilovich and Markovitch, 2006), since the topics are either not known in advance or are provided in the form of a controlled vocabulary with thousands of entries, and thus no classification can be performed. Instead, with topic identification, we aim to find topics (or categories[1]) that are relevant to the document at hand, which can be used to enrich the content of the document with relevant external knowledge.

2 Dynamic Ranking of Topic Relevance

Our method is based on the premise that external encyclopedic knowledge can be used to identify relevant topics for a given document.

The method consists of two main steps. In the first step, we build a knowledge graph of encyclopedic concepts based on Wikipedia, where the nodes in the graph are represented by the entities and categories that are defined in this encyclopedia. The edges between the nodes are represented by their relation of proximity inside the Wikipedia articles. The graph is built once and then it is stored offline, so that it can be efficiently use for the identification of topics in new documents.

In the second step, for each input document, we first identify the important encyclopedic concepts in the text, and thus create links between the content of the document and the external encyclopedic graph. Next, we run a biased graph centrality algorithm on the entire graph, so that all the nodes in the external knowledge repository are ranked based on their relevance to the input document. We use a variation

[1]Throughout the paper, we use the terms "topic" and "category" interchangeably.

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 210–218,
Boulder, Colorado, June 2009.

of the PageRank (Brin and Page, 1998) algorithm, which accounts for both the relation between the nodes in the document and the encyclopedic graph, as well as the relation between the nodes in the encyclopedic graph itself.

In the following, we first describe the structure of Wikipedia, followed by a brief description of the Wikify! system that automatically identifies the encyclopedic concepts in a text, and finally a description of the dynamic ranking process on the encyclopedic graph.

2.1 Wikipedia

Wikipedia (http://wikipedia.org) is a free online encyclopedia, representing the outcome of a continuous collaborative effort of a large number of volunteer contributors. Virtually any Internet user can create or edit a Wikipedia webpage, and this "freedom of contribution" has a positive impact on both the quantity (fast-growing number of articles) and the quality (potential mistakes are quickly corrected within the collaborative environment) of this resource.

Wikipedia has grown to become one of the largest online repositories of encyclopedic knowledge, with millions of articles available for a large number of languages. In fact, Wikipedia editions are available for more than 250 languages, with a number of entries varying from a few pages to close to three million articles per language.

The basic entry in Wikipedia is an *article* (or *page*), which defines an entity or an event, and consists of a hypertext document with hyperlinks to other pages within or outside Wikipedia. The role of the hyperlinks is to guide the reader to pages that provide additional information about the entities or events mentioned in an article. Each article in Wikipedia is uniquely referenced by an identifier, which consists of one or more words separated by spaces or underscores, and occasionally a parenthetical explanation. The current version of the English Wikipedia consists of about 2.75 million articles.

In addition to articles, Wikipedia also includes a large number of categories, which represent topics that are relevant to a given article (the July 2008 version of Wikipedia includes about 390,000 such categories). The category links are organized hierarchically, and vary from broad topics such as "history" or "games" to highly focused topics such as "military history of South Africa during World War II" or "role-playing game publishing companies."

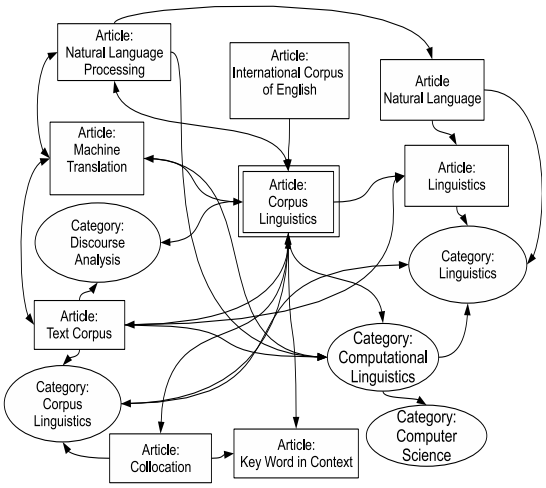


Figure 1: A snapshot from the encyclopedic graph.

We use the entire English Wikipedia to build an encyclopedic graph for use in the topic identification process. The nodes in the graph are represented by all the article and category pages in Wikipedia, and the edges between the nodes are represented by their relation of proximity inside the articles. The graph contains 5.8 million nodes, and 65.5 million edges. Figure 1 shows a small section of the knowledge graph, as built starting with the article on "Corpus Linguistics".

2.2 Wikify!

In order to automatically identify the important encyclopedic concepts in an input text, we use the unsupervised system Wikify! (Mihalcea and Csomai, 2007), which identifies the concepts in the text that are likely to be highly relevant (i.e., "keywords") for the input document, and links them to Wikipedia concepts.

Wikify! works in three steps, namely: (1) candidate extraction, (2) keyword ranking, and (3) word sense disambiguation. The candidate extraction step parses the input document and extracts all the possible n-grams that are also present in the vocabulary used in the encyclopedic graph (i.e., anchor texts for links inside Wikipedia or article or category titles).

Next, the ranking step assigns a numeric value to each candidate, reflecting the likelihood that a given candidate is a valuable keyword. Wikify! uses a "keyphraseness" measure to estimate the probability of a term W to be selected as a keyword in a

document, by counting the number of documents where the term was already selected as a keyword $count(D_{key})$ divided by the total number of documents where the term appeared $count(D_W)$. These counts are collected from all the Wikipedia articles.

$$P(keyword|W) \approx \frac{count(D_{key})}{count(D_W)} \quad (1)$$

This probability can be interpreted as "the more often a term was selected as a keyword among its total number of occurrences, the more likely it is that it will be selected again."

Finally, a simple word sense disambiguation method is applied, which identifies the most likely article in Wikipedia to which a concept should be linked to. This step is trivial for words or phrases that have only one corresponding article in Wikipedia, but it requires an explicit disambiguation step for those words or phrases that have multiple meanings (e.g., "plant") and thus multiple candidate pages to link to. The algorithm is based on statistical methods that identify the frequency of meanings in text, combined with symbolic methods that attempt to maximize the overlap between the current document and the candidate Wikipedia articles. See (Mihalcea and Csomai, 2007) for more details.

2.3 Biased Ranking of the Wikipedia Graph

Starting with the graph of encyclopedic knowledge, and knowing the nodes that belong to the input document, we want to rank all the nodes in the graph so that we obtain a score that indicates their importance relative to the given document. We can do this by using a graph-ranking algorithm *biased* toward the nodes belonging to the input document.

Graph-based ranking algorithms such as PageRank are a way of deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph. One formulation is in terms of a random walk through a directed graph. A "random surfer" visits nodes of the graph, and has some probability of jumping to some other random node of the graph, and the remaining probability of continuing their walk from the current node to one in its outdegree list. The rank of a node is an indication of the probability that the surfer would be found at that node at any given time.

Formally, let $G = (V, E)$ be a directed graph with the set of vertices V and set of edges E, where E is a subset of $V \times V$. For a given vertex V_i, let $In(V_i)$ be the set of vertices that point to it (predecessors), and let $Out(V_i)$ be the set of vertices that vertex V_i points to (successors). The score of a vertex V_i is defined as follows (Brin and Page, 1998):

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j) \quad (2)$$

where d is a damping factor usually set to 0.85.

Given the "random surfer" interpretation of the ranking process, the $(1-d)$ portion represents the probability that a surfer will jump to a given node from any other node at random, and the summation portion indicates that the process will enter the node via edges directly connected to it.

We introduce a bias in this graph-based ranking algorithm by extending the framework of personalization of PageRank proposed by (Haveliwala, 2002). We modify the formula so that the $(1-d)$ component also accounts for the importance of the concepts found in the input document, and it is suppressed for all the nodes that are not found in the input document.

$$S(V_i) = (1-d) * Bias(V_i) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j) \quad (3)$$

where $Bias(V_i)$ is only defined for those nodes initially identified in the input document:

$$Bias(V_i) = \frac{f(V_i)}{\sum_{j \in InitalNodeSet} f(V_j)}$$

and 0 for all other nodes in the graph. $InitalNodeSet$ is the set of nodes belonging to the input document.

Note that $f(V_i)$ can vary in complexity from a default value of 1 to a complex knowledge-based estimation. In our implementation, we use a combination of the "keyphraseness" score assigned to the node V_i and its distance from the "Fundamental" category in Wikipedia.

The use of the $Bias$ assigned to each node means the surfer random jumps will be limited to only those nodes connected to the original query. Thus the graph-ranking process becomes biased and focused on those topics directly related to the input. It also accumulates activation at those nodes not directly found in the input text, but linked through indirect means, thus reinforcing the nodes where patterns of activation intersect and creating a constructive interference pattern in the network. These reinforced nodes are the "implied related topics" of the text.

3 Illustration

To illustrate the ranking process, consider as an example the following sentence "The United States was involved in the Cold War."

First the text is passed through the Wikify! system, which returns the articles "United States" and "Cold War." Taking into account their "keyphraseness" as calculated by Wikify!, the selections are given an initial bias of 0.5492 ("United States") and 0.4508 ("Cold War").

After the first iteration the initial activation spreads out into the encyclopedic graph, the nodes find a direct connection to one another, and correspondingly their scores are changed to 0.3786 ("United States") and 0.3107 ("Cold War"). After the second iteration, new nodes are identified from the encyclopedic graph, a subset of which is shown in Figure2. The process will eventually continue for several iterations until the scores of the nodes do not change. The nodes with the highest scores in the final graph are considered to be the most closely related to the input sentence, and thus selected as relevant topics.

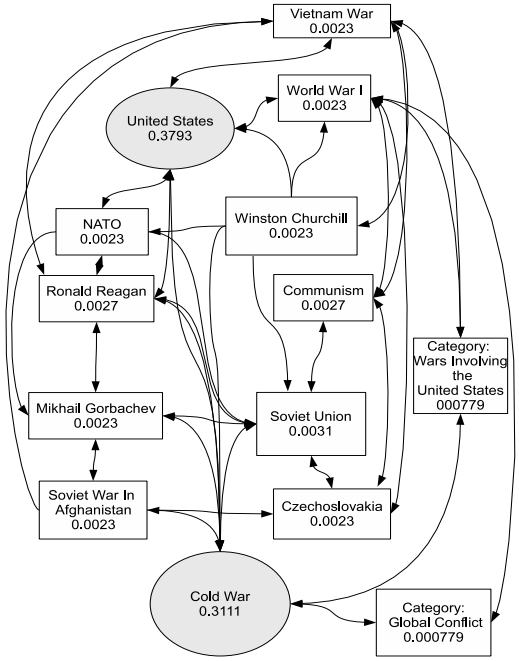


Figure 2: Sub-graph between "United States" and "Cold War"

In order to see the effect of the initial bias, consider as an example the ranking of the nodes in the encyclopedic graph when biased with the sentence "The United States was involved in the Cold War," versus the sentence "Microsoft applies Computer Science." A comparison between the scores of the nodes when activated by each of these sentences is shown in Table 1.

Wikipedia entry	US/CW	MS/CS	Diff.
A: United States	0.393636	0.006578	0.387058
C: Computer Science	0.000004	0.003576	-0.003571
A: World War II	0.007102	0.003674	0.003428
A: United Kingdom	0.005346	0.002670	0.002676
C: Microsoft	0.000001	0.001839	-0.001837
C: Cold War	0.001695	0.000006	0.001689
C: Living People	0.000835	0.002223	-0.001387
C: Mathematics	0.000029	0.001337	-0.001307
C: Computing	0.000008	0.001289	-0.001280
C: Computer Pioneers	0.000002	0.001238	-0.001235

Table 1: Node ranking differences when the encyclopedic graph is biased with different inputs: (1) "United States" and "Cold War" (US/CW) vs. (2) "Microsoft" and "Computer Science" (MS/CS). The nodes are either article pages (A) or category pages (C).

4 Experiments

In order to measure the effectiveness of the topic ranking process, we run three sets of experiments, aimed at measuring the relevancy of the automatically identified topics with respect to manually annotated gold standard data sets.

In the first experiment, the identification of the important concepts in the input text (used to bias the topic ranking process) is performed manually, by the Wikipedia users. In the second and third experiment, the identification of these important concepts is done automatically, by the Wikify! system. In all the experiments, the ranking of the concepts from the encyclopedic graph is done automatically by using the dynamic ranking process described in Section 2.

In the first two experiments, we use a data set consisting of 150 articles from Wikipedia, which have been explicitly removed from the encyclopedic graph. All the articles in this data set include manual annotations of the relevant categories, as assigned by the Wikipedia users, against which we can measure the quality of the automatic topic assignments. The 150 articles have been randomly selected while following the constraint that they each contain at least three article links and at least three category links. Our task is to rediscover the relevant categories for each page. Note that the task is nontrivial, since there are approximately 390,000 categories to choose from. We evaluate the quality of our system through the standard measures of preci-

sion and recall.

4.1 Manual Annotation of the Input Text

In this first experiment, the articles in the gold standard data set also include manual annotations of the important concepts in the text, i.e., the links to other Wikipedia articles as created by the Wikipedia users. Thus, in this experiment we only measure the accuracy of the dynamic topic ranking process, without interference from the Wikify! system.

There are two main parameters that can be set during a system run. First, the set of initial nodes used as bias in the ranking can include: (1) the initial set of articles linked to by the original document (via the Wikipedia links); (2) the categories listed in the articles linked to by the original document[2]; and (3) both. Second, the dynamic ranking process can be run through propagation on an encyclopedic graph that includes (1) all the articles from Wikipedia; (2) all the categories from Wikipedia; or (3) all the articles and the categories from Wikipedia.

Figures 3, 4 and 5 show the precision, recall and F-measure obtained for the various settings. In the plots, $Bias$ and $Propagate$ indicate the selections made for the two parameters, which can be each set to $Articles$, $Categories$, or $Both$. Each of these correspond to the options listed before.

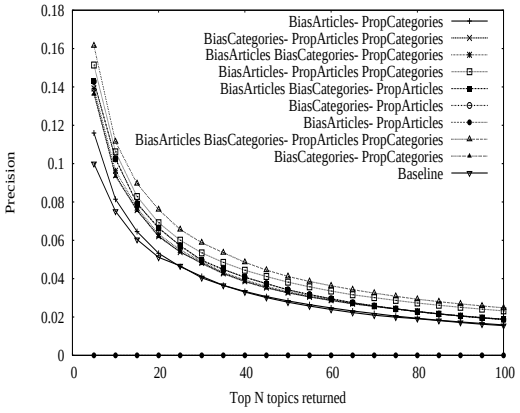


Figure 3: Precision for manual input text annotations.

As seen in the figures, the best results are obtained for a setting where both the initial bias and the propagation include all the available nodes, i.e., both articles and categories. Although the primary task is the identification of the categories, the addition of the article links improves the system performance.

[2]These should not be confused with the categories included in the document itself, which represent the gold standard annotations and are not used at any point.

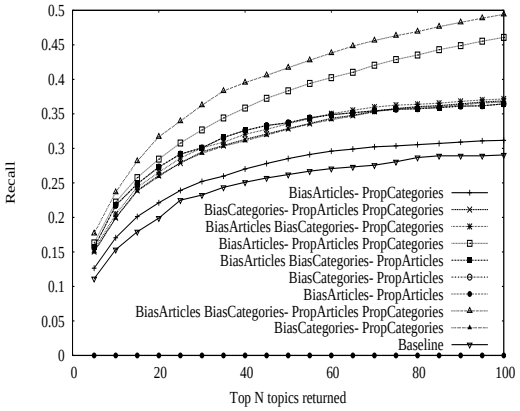


Figure 4: Recall for manual input text annotations.

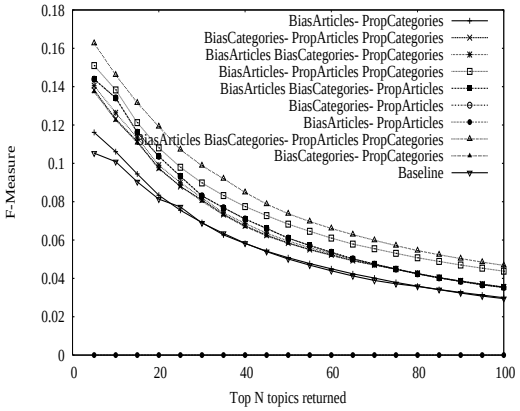


Figure 5: F-measure when using Wikipedia article annotations.

To place results in perspective, we also calculate a baseline (labeled as "Baseline" in the plots), which selects by default all the categories listed in the articles linked to by the original document. Each baseline article assigns $1/N$ to each of its N possible categories, with categories pointed to by multiple articles receiving the summation.

4.2 Automatic Annotation of the Input Text

The second experiment is similar to the first one, except that rather than using the manual annotations of the important concepts in the input document, we use instead the Wikify! system that automatically identifies these important concepts by using the method briefly described in Section 2.2. The article links identified by Wikify! are treated in the same way as the human anchor annotations from the previous experiment. In this experiment, we have

an additional parameter, which consists of the percentage of links selected by Wikify! out of the total number of words in the document. We refer to this parameter as keyRatio. The higher the keyRatio, the more terms are added, but also the higher the potential of noise due to mis-disambiguation.

Figures 6, 7 and 8 show the effect of varying the value of the keyRatio parameter used by Wikify! has on the precision, recall and F-measure of the system. Note that in this experiment, we only use the best setting for the other two parameters as identified in the previous experiment, namely an initial bias and a propagation step that include all available nodes, i.e., both articles and categories.

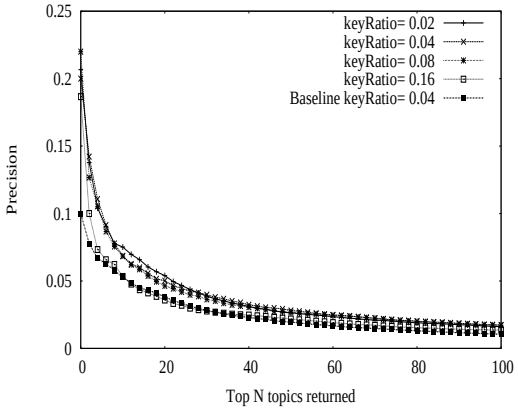


Figure 6: Precision for automatic input text annotations (Wikipedia data set)

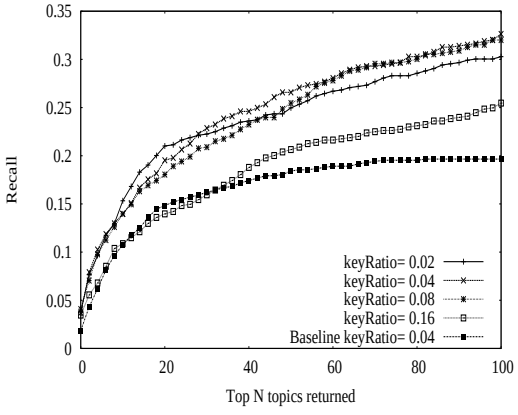


Figure 7: Recall for automatic input text annotations (Wikipedia data set)

The system's best performance occurs for a keyRatio of 0.04 to 0.06, which coincides with the ratio found optimal in previous experiments using the Wikify! system (Mihalcea and Csomai, 2007).

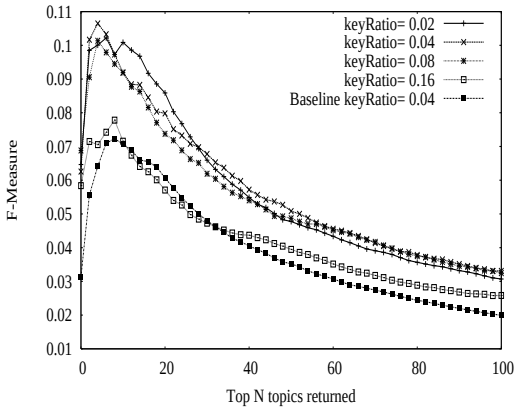


Figure 8: F-measure for automatic input text annotations (Wikipedia data set)

As before, we also calculate a baseline, which selects by default all the categories listed in the articles linked to by the original document, with the links being automatically identified with the Wikify! system. The baseline is calculated for a keyRatio of 0.04, which is one of the values that were found to work well for the ranking system itself and in previous Wikify! experiments.

Overall, the system manages to find many relevant topics for the documents in the evaluation data set, despite the large number of candidate topics (close to 390,000). Our system exceeds the baseline by a large margin, demonstrating the usefulness of using the biased ranking on the encyclopedic graph.

4.3 Article Selection for Computer Science Texts

In the third experiment, we use again the Wikify! system to annotate the input documents, but this time we run the evaluations on a data set consisting of computer science documents. We use the data set introduced in previous work on topic identification (Medelyan and Witten, 2008), where 20 documents in the field of computer science were independently annotated by 15 teams of two computer science undergraduates. The teams were asked to read the texts and assign to each of them the title of the five Wikipedia articles they thought were the most relevant **and** the other groups would also select. Thus, the consistency of the annotations was an important measure for this data set. (Medelyan and Witten, 2008) define consistency as a measure of agreement:

$$Consistency = \frac{2C}{A+B}$$

where A and B are the number of terms assigned by two indexing teams, and C is the number of terms they have in common. In the annotations experiments reported in (Medelyan and Witten, 2008), the human teams consistency ranged from 21.4% to 37.1%, with 30.5% being the average.[3]

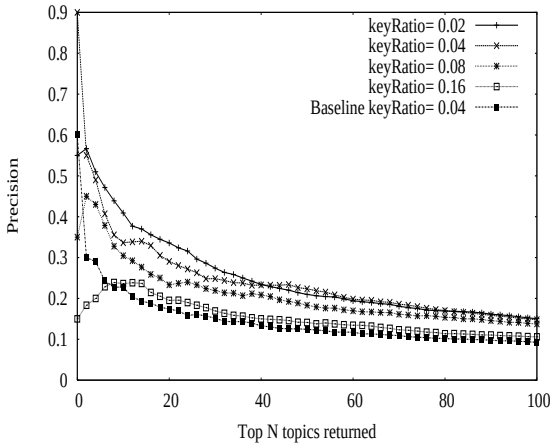


Figure 9: Precision for automatic input text annotations (Waikato data set)

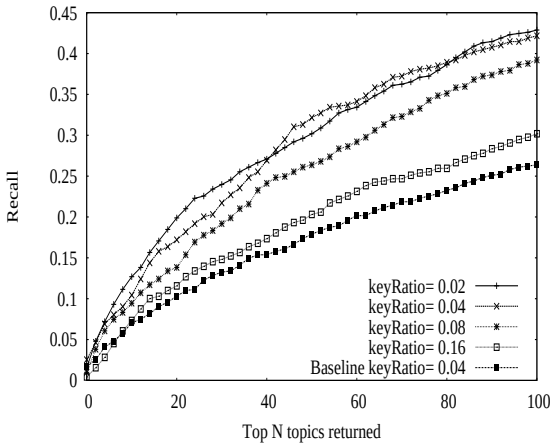


Figure 10: Recall for automatic input text annotations (Waikato data set)

Figures 10, 9, 11 and 12 show the performance of our system on this data set, by using the Wikify! annotations for the initial bias, and then propagating to both articles and categories. The plots also show a baseline that selects all the articles automatically identified in the original document by using the Wikify! system with a keyRatio set to 0.04.

[3]The consistency for one team is measured as the average of the consistencies with the remaining 14 teams.

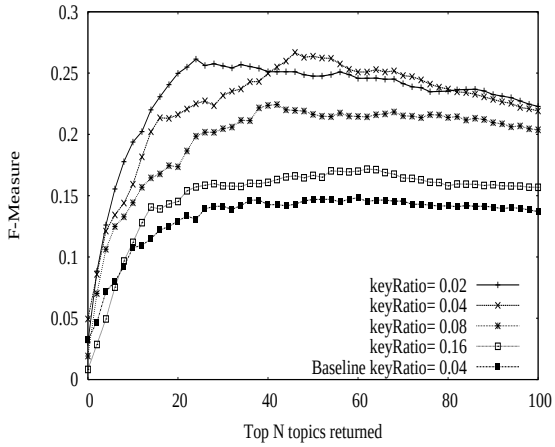


Figure 11: F-measure for automatic input text annotations (Waikato data set)

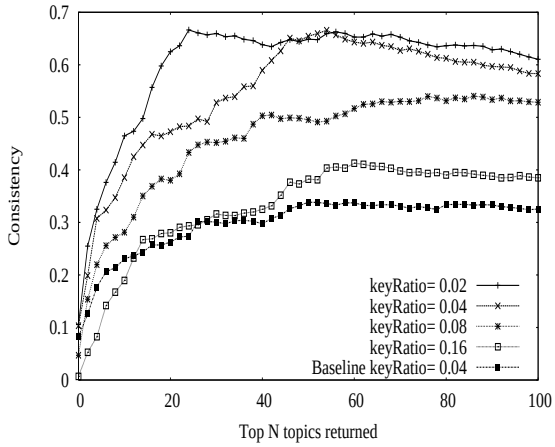


Figure 12: Consistency for automatic input text annotations (Waikato data set)

When selecting the top five topics returned by our system (the same number of topics as provided by the human teams), the average consistency with respect to the 15 human teams was measured at 34.5%, placing it between the 86% and 93% percentile of the human participants, with only two human teams doing better. We can compare this result with the one reported in previous work for the same data set. Using a machine learning system, (Medelyan and Witten, 2008) reported a consistency of 30.5%. Thus, our result of 34.5% is significantly better, despite the fact that our method is unsupervised.

In a second evaluation, we also considered the union of all the terms assigned by the 15 teams. On average, each document was assigned 35.5 different terms by the human teams. If allowed to provide more annotations, our system peaks with a con-

sistency of 66.6% for the top 25 topics returned. The system has the ability to identify possible relevant alternative topics using the comprehensive catalog of Wikipedia computer science articles and their possible associations. A human team may not necessarily consider all of the possibilities or even be aware that some of the articles, possibly known and used by the other teams, exist.

5 Related Work

The work closest to ours is perhaps the one described in (Medelyan and Witten, 2008), where topics relevant to a given document are automatically selected by using a machine learning system. Unlike our unsupervised approach, (Medelyan and Witten, 2008) learn what makes a good topic by training on previously annotated data.

Also related is the Wikify! system concerned with the automatic annotation of documents with Wikipedia links (Mihalcea and Csomai, 2007). However, Wikify! is purely extractive, and thus it cannot identify important topics or articles that are not explicitly mentioned in the input text.

Explicit semantic analysis (Gabrilovich and Markovitch, 2006) was also introduced as a way to determine the relevancy of the Wikipedia articles with respect to a given input text. The resulting vector however is extremely large, and while it was found useful for the task of text classification with a relatively small number of categories, it would be difficult to adapt for topic identification when the number of possible topics grows beyond the approximately 390,000 under consideration. In a similar line of work, (Bodo et al., 2007) examined the use of Wikipedia and latent semantic analysis for the purposes of text categorization, but reported negative results when used for the categorization of the Reuters-21578 dataset.

Others are exploring the use of graph propagation for deriving semantic information. (Hughes and Ramage, 2007) described the use of a biased PageRank over the WordNet graph to compute word pair semantic relatedness using the divergence of the probability values over the graph created by each word. (Ollivier and Senellart, 2007) describes a method to determine related Wikipedia article using a Markov chain derived value called the green measure. Differences exist between the PageRank based methods used as a baseline in their work and the method proposed here, since our system can use the content of the article, multiple starting points, and tighter control of the random jump probability via the bias value. Finally, (Syed et al., 2008) reported positive results by using various methods for topic prediction including the use of text similarity and spreading activation. The method was tested by using randomly selected Wikipedia articles, where in addition to the categories listed on a Wikipedia page, nearby subsuming categories were also included as acceptable.

6 Conclusions and Future Work

In this paper, we introduced a system for automatic topic identification, which relies on a biased graph centrality algorithm applied on a richly interconnected encyclopedic graph built from Wikipedia. Experiments showed that the integration of encyclopedic knowledge consistently adds useful information when compared to baselines that rely exclusively on the text at hand. In particular, when tested on a data set consisting of documents manually annotated with categories by Wikipedia users, the topics identified by our system were found useful as compared to the manual annotations. Moreover, in a second evaluation on a computer science data set, the system exceeded the performance of previously proposed machine learning algorithms, which is remarkable given the fact that our system is unsupervised. In terms of consistency with manual annotations, our system's performance was found to be comparable to human annotations, with only two out of 15 teams scoring better than our system.

The system provides a means to generate a dynamic ranking of topics in Wikipedia within a framework that has the potential to utilize knowledge or heuristics through additional resources (like ontologies) converted to graph form. This capability is not present in resources like search engines that provide access to a static ranking of Wikipedia. Future work will examine the integration of additional knowledge sources and the application of the method for metadata document annotations.

Acknowledgments

This work has been partially supported by an award #CR72105 from the Texas Higher Education Coordinating Board and by an award from Google Inc. The authors are grateful to the Waikato group for making their data set available.

References

Z. Bodo, Z. Minier, and L. Csato. 2007. Text categorization experiments using Wikipedia. In *Proceedings of the International Conference on Knowledge Engineering ,Principles and Techniques*, Cluj-Napoca (Romania).

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7).

E. Gabrilovich and S. Markovitch. 2006. Overcoming the brittleness bottleneck using Wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Boston.

T. Haveliwala. 2002. Topic-sensitive PageRank. In *Proceedings of the Eleventh International World Wide Web Conference*, May.

T. Hughes and D. Ramage. 2007. Lexical semantic knowledge with random graph walks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Prague, Czech Republic.

O. Medelyan and I. H. Witten. 2008. Topic indexing with Wikipedia. In *Proceedings of the AAAI WikiAI workshop*.

R. Mihalcea and A. Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, Lisbon, Portugal.

Y. Ollivier and P. Senellart. 2007. Finding related pages using green measures: An illustration with wikipedia. In *Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence (AAAI 2007)*.

Z. Syed, T. Finin, and A. Joshi. 2008. Wikipedia as an Ontology for Describing Documents. In *Proceedings of the Second International Conference on Weblogs and Social Media*. AAAI Press, March.

New Features for FrameNet – WordNet Mapping

Sara Tonelli and Daniele Pighin
FBK-Irst, Human Language Technologies
Via di Sommarive, 18 I-38100 Povo (TN) Italy
{satonelli,pighin}@fbk.eu

Abstract

Many applications in the context of natural language processing or information retrieval may be largely improved if they were able to fully exploit the rich semantic information annotated in high-quality, publicly available resources such as the FrameNet and the WordNet databases. Nevertheless, the practical use of similar resources is often biased by the limited coverage of semantic phenomena that they provide.

A natural solution to this problem would be to automatically establish anchors between these resources that would allow us 1) to jointly use the encoded information, thus possibly overcoming limitations of the individual corpora, and 2) to extend each resource coverage by exploiting the information encoded in the others.

In this paper, we present a supervised learning framework for the mapping of FrameNet lexical units onto WordNet synsets based on a reduced set of novel and semantically rich features. The automatically learnt mapping, which we call *MapNet*, can be used 1) to extend frame sets in the English FrameNet, 2) to populate frame sets in the Italian FrameNet via MultiWordNet and 3) to add frame labels to the MultiSemCor corpus. Our evaluation on these tasks shows that the proposed approach is viable and can result in accurate automatic annotations.

1 Introduction

In recent years, the integration of manually-built lexical resources into NLP systems has received growing interest. In particular, resources annotated with the surface realization of semantic roles, like FrameNet (Baker et al., 1998) or PropBank (Palmer et al., 2005) have shown to convey an improvement in several NLP tasks, from question answering (Shen and Lapata, 2007) to textual entailment (Burchardt et al., 2007) and shallow semantic parsing (Giuglea and Moschitti, 2006). Nonetheless, the main limitation of such resources is their poor coverage, particularly as regards FrameNet. Indeed, the latest FrameNet release (v. 1.3) contains 10,195 lexical units (LUs), 3,380 of which are described only by a lexicographic definition without any example sentence. In order to cope with this lack of data, it would be useful to map frame information onto other lexical resources with a broader coverage. We believe that WordNet (Fellbaum, 1998), with 210,000 entries in version 3.0, can represent a suitable resource for this task. In fact, both FrameNet and WordNet group together semantically similar words, and provide a hierarchical representation of the lexical knowledge (in WordNet the relations between synsets, in FrameNet between frames, see Ruppenhofer et al. (2006)). On the other hand, WordNet provides a more extensive coverage particularly for adjectives and nouns denoting artifacts and natural kinds, that are mostly neglected in FrameNet.

In this paper, we present an approach using Support Vector Machines (SVM) to map FrameNet lexical units to WordNet synsets. The proposed approach addresses some of the limitations of previous works on the same task (see for example De Cao et al. (2008) and Johansson and Nugues (2007)). Most notably, as we do not train the SVM on a per-

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL), pages 219–227,
Boulder, Colorado, June 2009.

frame basis, our model is able to cope also with those frames that have little or no annotated sentences to support the frame description. After learning a very fast model on a small set of annotated lexical unit-synset pairs, we can automatically establish new mappings in never-seen-before pairs and use them for our applications. We will evaluate the effect of the induced mappings on two tasks: the automatic enrichment of lexical unit sets in the English and Italian FrameNet via MultiWordNet (Pianta et al., 2002), and the annotation of the MultiSemCor corpus (Bentivogli and Pianta, 2005) with frame labels.

The discussion is structured as follows: in Section 2 we review the main characteristics of FrameNet and WordNet; in Section 3 we discuss previous attempts to establish a mapping between them; in Section 4 we describe our supervised approach to map lexical units onto synsets; Section 5 details the dataset that we employed for our experiments; Section 6 describes the novel features that we used to characterize the mapping; Section 7 details the results of our experiments; in Section 8 we apply the mapping to three resource annotation tasks; finally, in Section 9 we draw our conclusions.

2 FrameNet and WordNet

The FrameNet database (Baker et al. (1998), Fillmore et al. (2003)) is an English lexical resource based on the description of some prototypical situations, the *frames*, and the frame-evoking words or expressions associated to them, the *lexical units* (LU). Every frame corresponds to a scenario involving a set of participants, the *frame elements* (FEs), that are typically the semantic arguments shared by all LUs in a frame.

We report in Table 1 the information recorded in FrameNet for the CAUSE_TO_WAKE frame. In the first row there is the frame definition with the relevant frame elements, namely AGENT, CAUSE, SLEEPER and SLEEP_STATE. Then there is the list of all lexical units evoking the frame and the corresponding part of speech. Note that, differently from WordNet synsets, a frame can contain LUs with different PoS as well as antonymous words. In the last row, an example for each frame element is reported. The lexical unit is underlined, while the constituent bearing the FE label is written in italics. The FrameNet resource is corpus-based, i.e. every lexical unit should be instantiated by at least one example sentence. Besides, every lexical unit comes with a manual lexicographic definition. The latest database release contains 795 frame definitions and 10,195 lexical units, instantiated through approximately 140.000 example sentences. Despite this, the database shows coverage problems when exploited for NLP tasks, and is still being extended by the Berkeley group at ICSI.

Frame: CAUSE_TO_WAKE		
Def.	An AGENT or CAUSE causes a SLEEPER to transition from the SLEEP_STATE to wakeful consciousness.	
LUs	awaken.v, get_up.v, rouse.v, wake.v, wake_up.v singe.v, sizzle.v, stew.v	
FEs	AGENT	*We* tried to rouse Peter.
	CAUSE	*The rain* woke the children.
	SLEEPER	*Neighbors* were awakened by screams.
	SL_STATE	He woke Constance *from her doze*.

Table 1: Frame CAUSE_TO_WAKE

WordNet (Fellbaum, 1998) is a lexical resource for English based on psycholinguistics principles and developed at Princeton University. It has been conceived as a computational resource aimed at improving some drawbacks of traditional dictionaries such as the circularity of definitions and the ambiguity of sense references. At present, it covers the majority of nouns, verbs, adjectives and adverbs in the English language, organized in synonym sets called *synsets*, which correspond to concepts. WordNet also includes a rich set of semantic relations across concepts, such as hyponymy, entailment, antonymy, similar-to, etc. Each synset is encoded as a set of synomyms having the same part of speech and described by a definition or *gloss*. In some cases, one or more example sentences may also be reported. The Princeton English WordNet has also been augmented with domain labels (Magnini and Cavaglià, 2000) that group synsets into homogeneous clusters in order to reduce polysemy in the database.

We believe that mapping FrameNet LUs to WordNet synsets would have at least three different advantages: 1) for the English FrameNet, it would automatically increase the number of LUs for frame by

importing all synonyms from the mapped synset(s), and would allow to exploit the semantic and lexical relations in WordNet to enrich the information encoded in FrameNet. This would help coping with coverage problems and disambiguating the LU senses. 2) For WordNet, it would be possible to add a semantic layer between the synset level and the domain level represented by frame relations, and to enrich the synsets with a computational description of the situation they refer to together with the semantic roles involved. 3) Since frames are mostly defined at conceptual level, the FrameNet model is particularly suitable for cross-lingual induction (Boas, 2005). In this framework, the FrameNet-WordNet mapping could help modelling frame-based resources for new languages using minimal supervision. In fact, the availability of multilingual resources like *MultiWordNet* (Pianta et al., 2002) and *EuroWordNet* (Vossen, 1998) allows to easily populate frame sets for new languages with reduced human effort and near-manual quality by importing all lemmas from the mapped synsets.

3 Related work

Several experiments have been carried out to develop a FrameNet-WordNet mapping and test its applications. Shi and Mihalcea (2005) described a semi-automatic approach to exploit VerbNet as a bridge between FrameNet and WordNet for verbs, using synonym and hyponym relations and similarity between Levin's verb classes and FrameNet frames. Their mapping was used to develop a rule-based semantic parser (Shi and Mihalcea, 2004) as well as to detect target words and assign frames for verbs in an open text (Honnibal and Hawker, 2005).

Burchardt et al. (2005) presented a rule-based system for the assignment of FrameNet frames by way of a "detour via WordNet". They applied a WordNet-based WSD system to annotate lexical units in unseen texts with their contextually determined WordNet synsets and then exploited synonyms and hypernyms information to assign the best frame to the lexical units. The system was integrated into the SALSA RTE system for textual entailment (Burchardt et al., 2007) to cope with sparse-data problems in the automatic assignment of frame labels.

Johansson and Nugues (2007) created a feature representation for every WordNet lemma and used it to train an SVM classifier for each frame that tells whether a lemma belongs to the frame or not. The best-performing feature representation was built using the sequence of unique identifiers for each synset in its hypernym tree and weigthing the synsets according to their relative frequency in the SemCor corpus. They used the mapping in the Semeval-2007 task on frame-semantic structure extraction (Baker et al., 2007) in order to find target words in open text and assign frames.

Crespo and Buitelaar (2008) carried out an automatic mapping of medical-oriented frames to WordNet synsets applying a *Statistical Hypothesis Testing* to select synsets attached to a lexical unit that were statistically significant using a given reference corpus. The mapping obtained was used to expand Spanish FrameNet using *EuroWordNet* (Vossen, 1998) and evaluation was carried out on the Spanish lexical units obtained after mapping.

Given a set of lexical units, De Cao et al. (2008) propose a method to detect the set of suitable WordNet senses able to evoke a frame by applying a similarity function that exploits different WordNet information, namely conceptual density for nouns, synonymy and co-hyponymy for verbs and synonymy for adjectives. The mapping approach was applied also to LU induction for the English FrameNet and for Italian frames via MultiWordNet.

4 Problem formulation

Our objective is to be able to assign to every lexical unit *l*, belonging to a frame F_i defined in the FrameNet database, one or more WordNet senses that best express the meaning of *l*. More specifically, for every $l \in F_i$, we consider the set of all WordNet senses where *l* appears, *CandSet*, and then find the best WordNet sense(s) $best_s \subset$ *CandSet* that express the meaning of *l*.

For example, the lexical unit *rouse.v* belonging to the CAUSE_TO_WAKE frame, is defined in FrameNet as "bring out of sleep; awaken". Its *CandSet* comprises 4 senses[1]: 1# *bestir, rouse* (become active); 2# *rout_out, drive_out, force_out, rouse* (force or drive_out); #3 *agitate, rouse, turn_on, charge, com-*

[1]The gloss is reported between parenthesis

move, excite, charge_up (cause to be agitated, excited or roused); #4 *awaken, wake, waken, rouse, wake_up, arouse* (cause to become awake or conscious). In this example, $best_s = \{\#4\}$ for *rouse.v* in CAUSE_TO_WAKE.

We aim at creating a mapping system that can achieve a good accuracy also with poorly-documented lexical units and frames. In fact, we believe that under real-usage conditions, the automatic induction of LUs is typically required for frames with a smaller LU set, especially for those with only one element. In the FrameNet database (v. 1.3), 33 frames out of 720 are described only by one lexical unit, and 63 are described by two. Furthermore, more than 3,000 lexical units are characterized only by the lexicographic definition and are not provided with example sentences. For this reason, we suggest an approach that makes also use of usually unexploited information in the FrameNet database, namely the *definition* associated to every lexical unit, and disregards example sentences.

This is the main point of difference between our and some previous works, e.g. Johansson and Nugues (2007) and De Cao et al. (2008), where unsupervised approaches are proposed which strongly rely either on the number of lexical units in a frame or on the example sentences available for l in the FrameNet corpus. We claim that the relative short time necessary to annotate a small dataset of frame-synset pairs will result in a more reliable mapping system and, as a consequence, in consistent time savings when we actually try to use the mappings for some tasks. The ability to cope with different cases while retaining a good accuracy will allow to bootstrap the mapping process in many cases where other approaches would have failed due to lack of training data.

To this end, we can train a binary classifier that, given l and *CandSet*, for each pair $\langle l, s\rangle$, $s \in CandSet$, delivers a positive answer if $s \in best_s$, and a negative one otherwise. To follow on the previous example, for *rouse.v* we would have 4 classifier examples, i.e. the pairs $\langle$rouse.v,#1$\rangle$, $\langle$rouse.v,#2$\rangle$, $\langle$rouse.v,#3$\rangle$ and $\langle$rouse.v,#4$\rangle$. Of these, only the last would be considered a positive instance. As a learning framework, we decided to use SVMs due to their classification accuracy and robustness to noisy data (Vapnik, 1998).

5 Dataset description

In order to train and test the classifier, we created a gold standard by manually annotating 2,158 LU-synset pairs as positive or negative examples. We don't have data about inter-annotator agreement because the dataset was developed only by one annotator, but De Cao et al. (2008) report 0.90 as Cohen's Kappa computed over 192 LU-synset pairs for the same mapping task. This confirms that senses and lexical units are highly correlated and that the mapping is semantically motivated.

The annotation process can be carried out in reasonable time. It took approximately two work days to an expert annotator to manually annotate the 2,158 pairs that make up our gold standard. The lexical units were randomly selected from the FrameNet database regardless of their part of speech or amount of annotated data in the FrameNet database. For each lexical unit, we extracted from WordNet the synsets where the LU appears, and for each of them we assigned a positive label in case the LU-synset pairs share the same meaning, and a negative label otherwise. Statistics about the dataset are reported in Table 2.

N. of LU-synset pairs	2,158
N. of lexical units	617
Verbal lexical units	39%
Nominal lexical units	51%
Adjectival lexical units	9%
Adverbial lexical units	<1%
Targeted frames	386
Pairs annotated as positive	32%
Pairs annotated as negative	68%
Average polysemy	3.49
LUs with one candidate synset	204
LUs with 10 or more cand. synsets	32

Table 2: Statistics on the dataset

The 386 frames that are present in the dataset represent about one half of all lexicalized frames in the FrameNet database. This proves that, despite the limited size of the dataset, it is well representative of FrameNet characteristics. This is confirmed by the distribution of the part of speech. In fact, in the FrameNet database about 41% of the LUs

are nouns, 40% are verbs, 17% are adjectives and <1% are adverbs (the rest are prepositions, which are not included in our experiment because they are not present in WordNet). In our dataset, the percentage of nouns is higher, but the PoS ranking by frequency is the same, with nouns being the most frequent PoS and adverbs the less represented. The average polysemy corresponds to the average number of candidate synsets for every LU in the dataset. Note that the high number of lexical units with only one candidate does not imply a more straightforward mapping, because in some cases the only candidate represents a negative example. In fact, a LU could be encoded in a frame that does not correspond to the sense expressed by the synset.

6 Feature description

For every LU-synset pair in the gold standard, we extracted a set of features that characterize different aspects of the mapping. In the remainder, we detail the meaning as well as the feature extraction procedure of each of them.

Stem overlap Both WordNet glosses and LU definitions in FrameNet are manually written by lexicographers. We noticed that when they share the same sense, they show high similarity, and sometimes are even identical. For example, the definition of *thicken* in the *Change_of_consistency* frame is *"become thick or thicker"*, which is identical to the WordNet gloss of synset n. v#00300319. The *thicken* lemma occurs in three WordNet synsets, and in each of them it is the only lemma available, so no other information could be exploited for the sense disambiguation.

We believe that this information could help in the choice of the best candidate synset, so we stemmed all the words in the synset gloss and in the lexical unit definition and measured their overlap. As features, we use the ratio between the number of overlapping words and the number of words in the definition, both for the gloss and the LU description.

Prevalent Domain and Synset Since a frame represents a prototypical situation evoked by the set of its lexical units, our intuition is that it should be possible to assign it to a WordNet domain, that groups homogeneous clusters of semantically similar synsets (see Section 2).

Given the LU-synset pair $\langle l, s\rangle$, $l \in F_i$, $s \in CandSet$, we extract all the lexical units in F_i and then build a set *AllCandSet* of pairs $\langle s_j, c_j\rangle$, where s_j is a synset in which at least one $l_i \in F_i$ appears, and c_j is the count of lexical units that are found in s_j.

We exploit the information conveyed by *AllCandSet* in two ways: i) if there is a prevalent WordNet domain that characterizes the majority of the synsets in *AllCandSet*, and $s \in$ *CandSet* belongs to that same domain, we add a boolean feature to the feature vector representing $\langle l, s\rangle$; ii) if s is the synset with the highest count in *AllCandSet*, i.e. if $s = s_j$ and $c_j > c_i \forall \langle s_j, c_j\rangle \in AllCandSet$, $i \neq j$, then we add another boolean feature to encode this information.

Cross-lingual parallelism Our idea is that, if an English lexical unit and its Italian translation belong to the same frame, they are likely to appear also in the same MultiWordNet synset, and the latter would be a good candidate for mapping. In fact, in MultiWordNet the Italian WordNet is strictly aligned with the Princeton WordNet 1.6, with synsets having the same id for both languages, and also semantic relations are preserved in the multilingual hierarchy. Since no Italian FrameNet is available yet, we extended the parallel English-Italian corpus annotated on both sides with frame information described in Tonelli and Pianta (2008) by adding and annotating 400 new parallel sentences. The final corpus contains about 1,000 pairs of parallel sentences where the English and the Italian lexical unit belong to the same frame.

Given a pair $\langle l, s\rangle$, we check if l appears also in the corpus with the frame label F_i and extract its Italian translation l_{it}. If l_{it} appears also in the Italian version of synset s in MultiWordNet, we consider s as a good candidate for the mapping of l and encode this information as a binary feature.

Simple synset-frame overlap Intuitively, the more lemmas a frame and a synset have in common, the more semantically similar they are. In order to take into account this similarity in our feature vector, given the pair $\langle l, s\rangle$, $l \in F_i$, we extract all lexical units in F_i and all lemmas in s and we compute the number of overlapping elements. Then we divide

the value by the number of synsets where the same overlapping element(s) occur.

As an example, the words *tank* and *tank car* in the *Vehicle* frame, occur together only in the fourth synset related to *tank*, which therefore will have a higher value for this feature.

Extended synset-frame overlap This feature is a generalization of overlapping value described above. In fact, we noticed that the hypernym information in WordNet can help disambiguating the synsets. Therefore, we take into account not only the overlaps according to the previous criterion, but also the number of overlapping words between the lexical units in a frame and the hypernyms of a synset. For example, the *party.n* lexical unit in the AGGREGATE frame has 5 senses in WordNet. According to the previous criterion, there is no overlap between the LUs in the frame and the lemmas in any of the five synsets. Instead, if we look at the direct hypernym relation of *party*, we find that sense #3 is also described as *set, circle, band*, that are also lexical units of AGGREGATE.

In those cases where the hypernym relation is not defined, e.g. adjectives, we used the *similar-to* relation.

7 Experimental setup and evaluation

To evaluate our methodology we carried out a 10-fold cross validation using the available data, splitting them in 10 non-overlapping sets. For each iteration, 70% of the data was used for training, 30% for testing. All the splits were generated so as to maintain a balance between positive and negative examples in the training and test sets.

We used the SVM optimizer SVM-Light[2] (Joachims, 1999), and applied polynomial kernels (*poly*) of different degrees (i.e. 1 through 4) in order to select the configuration with the best generalization capabilities. The accuracy is measured in terms of Precision, Recall and F_1 measure, i.e. the harmonic average between Precision and Recall. For the sake of annotation, it is important that an automatic system be very precise, thus not producing wrong annotations. On the other hand, the higher the recall, the larger the amount of data that the system will be able to annotate.

[2]Available at http://svmlight.joachims.org/

The macro-average of the classifier accuracy for the different configurations is shown in Table 3. We report results for linear kernel (i.e. poly 1), maximizing recall and f-measure, and for polynomial kernel of degree 2 (i.e. poly 2), scoring the highest precision. In general , we notice that all our models have a higher precision than recall, but overall are quite balanced. Different polynomial kernels (i.e. conjunction of features) do not produce very relevant differences in the results, suggesting that the features that we employed encode significant information and have a relevance if considered independently.

As a comparison, we also carried out the same evaluation by setting a manual threshold and considering a LU-synset pair as a positive example if the sum of the feature values was above the threshold. We chose two different threshold values, the first (Row 1 in Table 3) selected so as to have comparable precision with the most precise SVM model (i.e. poly2), the second (Row 2) selected to have recall comparable with poly1, i.e. the SVM model with highest recall. In the former case, the model has a recall that is less than half than poly2, i.e. 0.214 vs. 0.569, meaning that such model would establish a half of the mappings while making the same percentage of mistakes. In the latter, the precision of the SVM classifier is 0.114 points higher, i.e. 0.794 vs. 0.680, meaning the SVM can retrieve as many mappings but making 15% less errors.

In order to investigate the impact of different features on the classifier performance, we also considered three different groups of features separately: the ones based on stem overlap, those computed for prevalent domain and synset, and the features for simple and extended frame – synset overlap. We did not take into account cross-lingual parallelism because it is one single feature whose coverage strongly relies on the parallel corpus available. As a consequence, it is not possible to test the feature in isolation due to data sparseness.

Results are shown in Table 3, in the second group of rows. Also in this case, we carried out a 10-fold cross validation using a polynomial kernel of degree 2. The stem overlap features, which to our best knowledge are an original contribution of our approach, score the highest recall among the three groups. This confirms our intuition that LU defini-

tions and WordNet glosses can help extending the number of mapped LUs, including those that are poorly annotated. For instance, if we consider the KNOT_CREATION frame, having only *tie.v* as LU, the features about prevalent domain & synset and about synset-frame overlap would hardly be informative, while stem overlap generally achieves a consistent performance regardless of the LU set. In fact, *tie.v* is correctly mapped to synset v#00095054 based on their similar definition (respectively *"to form a knot"* and *"form a knot or bow in"*). Best precision was scored by the feature group considering prevalent domain & synset, which are also new features introduced by our approach. The positive effect of combining all features is clearly shown by comparing the results obtained with individual feature groups against the figures in the row labeled *poly2*.

	Prec.	Recall	F1
Man. thresh. (P)	0.789	0.214	0.337
Man. thresh. (F1)	0.680	0.662	0.671
Stem Overlap	0.679	0.487	0.567
Prev.Dom.& Syn.	0.756	0.434	0.551
Syn.- Frame Overlap	0.717	0.388	0.504
poly1	0.761	**0.613**	**0.679**
poly2	**0.794**	0.569	0.663

Table 3: Mapping evaluation

8 MapNet and its applications

Since we aim at assigning at least one synset to every lexical unit in FrameNet, we considered all the frames and for every LU in the database we created a list of LU-synset pairs. We re-trained the classifier using the whole annotated gold standard and classified all the candidate pairs. The mapping produced between the two resources, that we call *MapNet*, comprises 5,162 pairs. Statistics on MapNet are reported in table 4.

About one thousand lexical units in FrameNet have no candidate synsets because the lemma is not present in WordNet. The remaining LUs have 3.69 candidate synsets each on average, similarly to the average polysemy reported for the gold standard (see Table 2). This confirms our hypothesis that the data used for training are well representative of the characteristics of the whole resource. We expect about 80% of these mappings to be correct, i.e. in line with the precision of the classifier.

N. of LUs in FrameNet	10,100
N. of LUs with at least one syn.cand.	9,120
N. of LU-synset candidate pairs	33,698
N. of mapped pairs	5,162

Table 4: Statistics on the mapping

8.1 Automatic FrameNet extension

MapNet can be easily exploited to automatically extend FrameNet coverage, in particular to extend the set of lexical units for each frame. In fact, we can assume that all lemmas in the mapped synsets have the same meaning of the LUs in the corresponding frames. We use MapNet to extract from WordNet the lemmas in the mapped synsets and add them to the frames.

For English FrameNet, we can acquire 4,265 new lexical units for 521 frames. In this way, we would extend FrameNet size by almost 42%. In the random evaluation of 100 newly acquired LUs belonging to 100 different frames, we assessed a precision of 78%. For the Italian side, we extract 6,429 lexical units for 561 frames. Since no Italian FrameNet has been developed yet, this would represent a first attempt to create this resource by automatically populating the frames. We evaluate the content of 15 complete frames containing 191 Italian LUs. The assigned LUs are correct in 88% of the considered cases, which represent a promising result w.r.t. the unsupervised creation of Italian FrameNet.

The difference in the evaluation for the two languages most likely lies in the smaller number of synsets on the Italian side of MultiWordNet if compared to the English, which results in less ambiguity. Furthermore, we should consider that the task for Italian is easier than for English, since in the former case we are building a resource from scratch, while in the latter we are extending an already existing resource with lexical units which are most likely peripheral with respect to those already present in the database.

8.2 Frame annotation of MultiSemCor

MultiSemCor (Bentivogli and Pianta, 2005) is an English/Italian parallel corpus, aligned at word level and annotated with PoS, lemma and WordNet synsets. The parallel corpus was created starting from the SemCor corpus, which is a subset of the English Brown corpus containing about 700,000 running words. The corpus was first manually translated into Italian. Then, the procedure of transferring word sense annotations from English to Italian was carried out automatically.

We apply *MapNet* to enrich the corpus with frame information. We believe that this procedure would be interesting from different point of views. Not only we would enrich the resource with a new annotation layer, but we would also automatically acquire a large set of English and Italian sentences having a lexical unit with a frame label. For the English side, it is a good solution to automatically extract a dataset with frame information and train, for example, a machine learning system for frame identification. For the Italian side, it represents a good starting point for the creation of a large annotated corpus with frame information, the base for a future Italian FrameNet.

MultiSemCor contains 12,843 parallel sentences. If we apply *MapNet* to the corpus, we produce 27,793 annotated instances in English and 23,872 in Italian, i.e. about two lexical units per sentence. The different amount of annotated sentences depends on the fact that in MultiSemCor some synset annotations have not been transferred from English to Italian. From both sides of the resulting corpus, we randomly selected 200 sentences labeled with 200 different frames, and evaluated the annotation quality. As for the English corpus, 75% of the sentences was annotated with the correct frame label, while on the Italian side they were 70%. This result is in line with the expectations, since *MapNet* was developed with 0.79 precision. Besides, synset annotation on the English side of MultiSemCor was carried out by hand, while annotation in Italian was automatically acquired by transferring the information from the English corpus (precision 0.86). This explains why the resulting annotation for English is slightly better than for Italian. In some cases, the wrongly annotated frame was strictly connected to the right one, i.e. APPLY_HEAT instead of COOKING_CREATION and ATTACHING instead of INCHOATIVE_ATTACHING.

9 Conclusions

We proposed a new method to map FrameNet LUs to WordNet synsets using SVM with minimal supervision effort.

To our best knowledge, this is the only approach to the task that exploits features based on stem overlap between LU definition and synset gloss and that makes use of information about WordNet domains. Differently from other models, the SVM is not trained on a per-frame basis and we do not rely on the number of the annotated sentences for a LU in the FrameNet corpus, thus our mapping algorithm performs well also with poorly-annotated LUs. After creating *MapNet*, the mapping between FrameNet and WordNet, we applied it to three tasks: the automatic induction of new LUs for English FrameNet, the population of frames for Italian FrameNet and the annotation of the MultiSemCor corpus with frame information. A preliminary evaluation shows that the mapping can significantly reduce the manual effort for the development and the extension of FrameNet-like resources, both in the phase of corpus annotation and of frame population.

In the future, we plan to improve the algorithm by introducing syntactic features for assessing similarity between LU definitions and WordNet glosses. We also want to merge all information extracted and collected for Italian FrameNet and deliver a seed version of the resource to be validated. Finally, we plan to extend the mapping to all languages included in MultiWordNet, i.e. Spanish, Portuguese, Hebrew and Romanian.

Acknowledgements

We thank Roberto Basili and Diego De Cao for sharing with us their gold standard of frame – synset mappings.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th ACL Meeting and 17th ICCL Conference*. Morgan Kaufmann.

Collin F. Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval-2007 Task 10: Frame Semantic Structure Extraction. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 99–104, Prague, CZ, June.

Luisa Bentivogli and Emanuele Pianta. 2005. Exploiting Parallel Texts in the Creation of Multilingual Semantically Annotated Resources: The MultiSemCor Corpus. *Natural Language Engineering, Special Issue on Parallel Texts*, 11(03):247–261, September.

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

Aljoscha Burchardt, Katrin Erk, and Annette Frank. 2005. A WordNet detour to FrameNet. In B. Fisseni, H. Schmitz, B. Schröder, and P. Wagner, editors, *Sprachtechnologie, mobile Kommunikation und lingistische Resourcen*, Frankfurt am Main, Germany. Peter Lang.

Aljoscha Burchardt, Nils Reiter, Stefan Thater, and Annette Frank. 2007. A semantic approach to textual entailment: System evaluation and task analysis. In *Proceedings of Pascal RTE-3 Challenge*, Prague, CZ.

Diego De Cao, Danilo Croce, Marco Pennacchiotti, and Roberto Basili. 2008. Combining Word Sense and Usage for modeling Frame Semantics. In *Proceedings of STEP 2008*, Venice, Italy.

Mario Crespo and Paul Buitelaar. 2008. Domain-specific English-to-Spanish Translation of FrameNet. In *Proc. of LREC 2008*, Marrakech.

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

C.J. Fillmore, C.R. Johnson, and M. R. L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16:235–250, September.

Ana-Maria Giuglea and Alessandro Moschitti. 2006. Semantic role labeling via FrameNet, VerbNet and PropBank. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual ACL meeting*, pages 929–936, Morristown, NJ, US. Association for Computational Linguistics.

Matthew Honnibal and Tobias Hawker. 2005. Identifying FrameNet frames for verbs from a real-text corpus. In *Proceedings of Australasian Language Technology Workshop 2005*.

Thorsten Joachims. 1999. Making large-scale support vector machine learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J Smola, editors, *Advances in kernel methods: support vector learning*, pages 169–184. MIT Press, Cambridge, MA, USA.

R. Johansson and P. Nugues. 2007. Using WordNet to extend FrameNet coverage. In *Proc. of the Workshop on Building Frame-semantic Resources for Scandinavian and Baltic Languages, at NODALIDA*, Tartu.

Bernardo Magnini and Gabriela Cavaglià. 2000. Integrating Subject Field Codes into WordNet. In *Proceedings of LREC 2000*, pages 1413–1418, Athens, Greece.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31.

Emanuele Pianta, Luisa Bentivogli, and Christian Girardi. 2002. MultiWordNet: developing an aligned multilingual database. In *First International Conference on Global WordNet*, pages 292–302, Mysore, India.

Josef Ruppenhofer, Michael Ellsworth, Miriam R.L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2006. *FrameNet II: Extended Theory and Practice*. Available at http://framenet.icsi.berkeley.edu/book/book.html.

Dan Shen and Mirella Lapata. 2007. Using Semantic Roles to Improve Question Answering. In *Proceedings of EMNLP and CONLL*, pages 12–21, Prague, CZ.

Lei Shi and Rada Mihalcea. 2004. Open Text Semantic Parsing Using FrameNet and WordNet. In *Proceedings of HLT-NAACL 2004*.

Lei Shi and Rada Mihalcea. 2005. Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing. In *Proceedings of CICLing 2005*, pages 100–111. Springer.

Sara Tonelli and Emanuele Pianta. 2008. Frame Information Transfer from English to Italian. In European Language Resources Association (ELRA), editor, *Proceedings of LREC 2008*, Marrakech, Morocco.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

Piek Vossen, editor. 1998. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Springer, October.

Author Index