

CoNLL 2008

**Proceedings of the
Twelfth Conference on Computational
Natural Language Learning**

Conference chairs:
Alexander Clark and Kristina Toutanova

16–17 August 2008
Manchester, UK

©2008 The Coling 2008 Organizing Committee

Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Nonported* license
<http://creativecommons.org/licenses/by-nc-sa/3.0/>
Some rights reserved

Order copies of this and other Coling 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 978-1-905593-48-4

Design by Chimney Design, Brighton, UK
Production and manufacture by One Digital, Brighton, UK

Introduction

The 2008 Conference on Computational Natural Language Learning is the twelfth in the series of yearly meetings organized by SIGNLL, the ACL special interest group on natural language learning. CoNLL 2008 will be held in Manchester, UK, August 16-17, 2008, in conjunction with Coling 2008.

We are delighted to report that CoNLL's main session received a large number of submissions. A total of 85 papers were under consideration for the main session after several withdrawals, and of them only 20 were accepted. This makes this year's CoNLL especially competitive and contributes to an interesting program. We are grateful to the program committee members for their service in evaluating the submissions. Special thanks to the program committee members who joined on a short notice to help with the larger than expected number of submissions.

This year CoNLL had two special themes of interest, both of which solicited papers on models that explain natural phenomena relating to human language. The first concerned the central scientific problem addressed by CoNLL: the study of first language acquisition. The second theme was the central engineering problem: how to build systems that do something useful, especially complete systems that solve real problems.

The first theme contributed to an increased number of high-quality submissions in the first language acquisition area. Two sessions of the conference will be devoted to papers on this topic. The second theme led to submissions in diverse traditional NLP application areas.

As in previous years, CoNLL 2008 has a shared task. This year, the conference shared task proposed to merge the shared task topics from the last four years (2004-2007) into a unique task called "Joint Learning of Syntactic and Semantic Dependencies". Both syntactic dependencies (extracted from the Penn Treebank) and semantic dependencies (extracted from PropBank and NomBank) were jointly addressed under a unique unified representation.

The shared task was organized by Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre.

The call was very successful attracting the interest of more than 50 teams from all over the world, which represented a wide variety of universities, research institutions, and companies. At the end of the evaluation period, 22 teams submitted results (with 19 and 5 contributions to the closed and open challenges, respectively). All this work will be presented in the conference in the form of 5 selected oral talks and 14 posters.

In our opinion, the current shared task constitutes a qualitative step ahead and we hope that the resources created and the body of work presented will both serve as a benchmark and have a substantial impact on future research on syntactic-semantic parsing.

We are excited that the invited speakers at CoNLL 2008 will be Regina Barzilay and Nick Chater.

Finally, we would like to thank the SIGNLL board members for useful discussion, Erik Tjong Kim Sang, who acted as the information officer, and especially Lluís Màrquez and Joakim Nivre, who helped us greatly with advice around the conference organization, as well as to the organizers of COLING

2008, Harold Somers, Mark Stevenson and Roger Evans. Many thanks also to Microsoft Research for sponsoring CoNLL this year and to Priscilla Rasmussen for help with the finances.

Enjoy this year's conference!

Alex Clark and Kristina Toutanova

CoNLL 2008 Conference Chairs

Conference Chairs:

Alex Clark, Royal Holloway, University of London
Kristina Toutanova, Microsoft Research

Programme Committee:

Steve Abney, University of Michigan
Eneko Agirre, University of the Basque Country
Galen Andrew, Microsoft Research
Tim Baldwin, University of Melbourne
Regina Barzilay, MIT
Roberto Basili, University of Roma, Tor Vergata
Phil Blunsom, University of Edinburgh
Thorsten Brants, Google Research
Paula Buttery, Cambridge University
Xavier Carreras, MIT
Nick Chater, University College London
Ciprian Chelba, Google Research
Colin Cherry, Microsoft Research
Alexander Clark, Royal Holloway, University of London
Stephen Clark, Oxford University
James Cussens, University of York
Walter Daelemans, CNTS, Antwerp
Hal Daumé III, University of Utah
Jenny Finkel, Stanford University
Radu Florian, IBM research
Dayne Freitag, HNC Software
Michel Galley, Stanford University
Jianfeng Gao, Microsoft Research
Daniel Gildea, Rochester
Sharon Goldwater, University of Edinburgh
Jan Hajic, Institute of Formal and Applied Linguistics, Charles University in Prague
Marti Hearst, UC Berkeley
James Henderson, University of Geneva
Liang Huang, University of Pennsylvania
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

Roger Levy, UC San Diego
Percy Liang, UC Berkeley
Rob Malouf, San Diego State University
Lluís Màrquez, Technical University of Catalonia
Yuji Matsumoto, Nara Institute of Science and Technology
Diana McCarthy, Sussex University
Rada Mihalcea, University of North Texas
Alessandro Moschitti, DISI, University of Trento
John Nerbonne, University of Groningen
Hwee Tou Ng, National University of Singapore
Vincent Ng, University of Texas at Dallas
Joakim Nivre, Uppsala University
Franz Och, Google Research
Miles Osborne, University of Edinburgh
Slav Petrov, UC Berkeley
David Powers, Flinders University
Chris Quirk, Microsoft Research
Ari Rappoport, Hebrew University
Ellen Riloff, University of Utah
Dan Roth, University of Illinois, Urbana-Champaign
William Sakas, City University of New York
Anoop Sarkar, Simon Fraser University
Richard Sproat, University of Illinois, Urbana-Champaign
Suzanne Stevenson, University of Toronto
Mihai Surdeanu, Barcelona Media Information Center
Charles Sutton, UC Berkeley
Kristina Toutanova, Microsoft Research
Antal van den Bosch, Tilburg University
Charles Yang, University of Pennsylvania

Invited Speakers:

Regina Barzilay, MIT Computer Science & Artificial Intelligence Lab
Nick Chater, Department of Psychology, University College London

Table of Contents

<i>Semantic Parsing for High-Precision Semantic Role Labelling</i> Paola Merlo and Gabriele Musillo	1
<i>TAG, Dynamic Programming, and the Perceptron for Efficient, Feature-Rich Parsing</i> Xavier Carreras, Michael Collins and Terry Koo	9
<i>A Fast Boosting-based Learner for Feature-Rich Tagging and Chunking</i> Tomoya Iwakura and Seishi Okamoto	17
<i>Linguistic features in data-driven dependency parsing</i> Lilja Ovreliid	25
<i>Transforming Meaning Representation Grammars to Improve Semantic Parsing</i> Rohit Kate	33
<i>Using LDA to detect semantically incoherent documents</i> Hemant Misra, Olivier Cappe and Francois Yvon	41
<i>Picking them up and Figuring them out: Verb-Particle Constructions, Noise and Idiomaticity</i> Carlos Ramisch, Aline Villavicencio, Leonardo Moura and Marco Idiart	49
<i>Fast Mapping in Word Learning: What Probabilities Tell Us</i> Afra Alishahi, Afsaneh Fazly and Suzanne Stevenson	57
<i>Improving Word Segmentation by Simultaneously Learning Phonotactics</i> Daniel Blanchard and Jeffrey Heinz	65
<i>A MDL-based Model of Gender Knowledge Acquisition</i> Harmony Marchal, Benoît Lemaire, Maryse Bianco and Philippe Dessus	73
<i>Baby SRL: Modeling Early Language Acquisition.</i> Michael Connor, Yael Gertner, Cynthia Fisher and Dan Roth	81
<i>An Incremental Bayesian Model for Learning Syntactic Categories</i> Christopher Parisien, Afsaneh Fazly and Suzanne Stevenson	89
<i>Fully Unsupervised Graph-Based Discovery of General-Specific Noun Relationships from Web Corpora Frequency Counts</i> Gaël Dias, Raycho Mukelov and Guillaume Cleuziou	97
<i>Acquiring Knowledge from the Web to be used as Selectors for Noun Sense Disambiguation</i> Hansen A. Schwartz and Fernando Gomez	105
<i>Automatic Chinese Catchword Extraction Based on Time Series Analysis</i> Han Ren, Donghong Ji, Jing Wan and Lei Han	113
<i>Easy as ABC? Facilitating Pictorial Communication via Semantically Enhanced Layout</i> Andrew B. Goldberg, Xiaojin Zhu, Charles R. Dyer, Mohamed Eldawy and Lijie Heng	119
<i>A Nearest-Neighbor Approach to the Automatic Analysis of Ancient Greek Morphology</i> John Lee	127

<i>Context-based Arabic Morphological Analysis for Machine Translation</i> Thuy Linh Nguyen and Stephan Vogel	135
<i>A Tree-to-String Phrase-based Model for Statistical Machine Translation</i> Thai Phuong Nguyen, Akira Shimazu, Tu Bao Ho, Minh Le Nguyen and Vinh Van Nguyen ...	143
<i>Trainable Speaker-Based Referring Expression Generation</i> Giuseppe Di Fabbrizio, Amanda Stent and Srinivas Bangalore	151
<i>The CoNLL 2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies</i> Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez and Joakim Nivre	159
<i>A Latent Variable Model of Synchronous Parsing for Syntactic and Semantic Dependencies</i> James Henderson, Paola Merlo, Gabriele Musillo and Ivan Titov	178
<i>Dependency-based Syntactic–Semantic Analysis with PropBank and NomBank</i> Richard Johansson and Pierre Nugues	183
<i>A Joint Model for Parsing Syntactic and Semantic Dependencies</i> Xavier Lluís and Lluís Màrquez	188
<i>Collective Semantic Role Labelling with Markov Logic</i> Sebastian Riedel and Ivan Meza-Ruiz	193
<i>Hybrid Learning of Dependency Structures from Heterogeneous Linguistic Resources</i> Yi Zhang, Rui Wang and Hans Uszkoreit	198
<i>Parsing Syntactic and Semantic Dependencies with Two Single-Stage Maximum Entropy Models</i> Hai Zhao and Chunyu Kit	203
<i>A Combined Memory-Based Semantic Role Labeler of English</i> Roser Morante, Walter Daelemans and Vincent Van Asch	208
<i>A Puristic Approach for Joint Dependency Parsing and Semantic Role Labeling</i> Alexander Volokh and Günter Neumann	213
<i>Discriminative Learning of Syntactic and Semantic Dependencies</i> Lu Li, Shixi Fan, Xuan Wang and Xiaolong Wang	218
<i>Discriminative vs. Generative Approaches in Semantic Role Labeling</i> Deniz Yuret, Mehmet Ali Yatbaz and Ahmet Engin Ural	223
<i>A Pipeline Approach for Syntactic and Semantic Dependency Parsing</i> Yotaro Watanabe, Masakazu Iwatate, Masayuki Asahara and Yuji Matsumoto	228
<i>Semantic Dependency Parsing using N-best Semantic Role Sequences and Roleset Information</i> Joo-Young Lee, Han-Cheol Cho and Hae-Chang Rim	233
<i>A Cascaded Syntactic and Semantic Dependency Parsing System</i> Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu and Sheng Li ...	238
<i>The Integration of Dependency Relation Classification and Semantic Role Labeling Using Bilayer Maximum Entropy Markov Models</i> Weiwei Sun, Hongzhan Li and Zhifang Sui	243

<i>Mixing and Blending Syntactic and Semantic Dependencies</i> Yvonne Samuelsson, Oscar Täckström, Sumithra Velupillai, Johan Eklund, Mark Fishel and Markus Saers	248
<i>Dependency Tree-based SRL with Proper Pruning and Extensive Feature Engineering</i> Hongling Wang, Honglin Wang, Guodong Zhou and Qiaoming Zhu	253
<i>DeSRL: A Linear-Time Semantic Role Labeling System</i> Massimiliano Ciaramita, Giuseppe Attardi, Felice Dell’Orletta and Mihai Surdeanu	258
<i>Probabilistic Model for Syntactic and Semantic Dependency Parsing</i> Enhong Chen, Liu Shi and Dawei Hu	263
<i>Applying Sentence Simplification to the CoNLL-2008 Shared Task</i> David Vickrey and Daphne Koller	268

Conference Programme

Saturday, August 16, 2008

8:30–8:50 Opening Remarks

Session 1: Parsing

8:50–9:15 *Semantic Parsing for High-Precision Semantic Role Labelling*
Paola Merlo and Gabriele Musillo

9:15–9:40 *TAG, Dynamic Programming, and the Perceptron for Efficient, Feature-Rich Parsing*
Xavier Carreras, Michael Collins and Terry Koo

9:40–10:05 *A Fast Boosting-based Learner for Feature-Rich Tagging and Chunking*
Tomoya Iwakura and Seishi Okamoto

10:05–10:30 *Linguistic features in data-driven dependency parsing*
Lilja Øvrelid

10:30–11:00 Coffee break

Session 2: Semantics

11:00–11:25 *Transforming Meaning Representation Grammars to Improve Semantic Parsing*
Rohit Kate

11:25–11:50 *Using LDA to detect semantically incoherent documents*
Hemant Misra, Olivier Cappe and Francois Yvon

11:50–12:40 Invited talk by Regina Barzilay

12:40–14:00 Lunch

Saturday, August 16, 2008 (continued)

Shared Task

- 14:00–14:30 *The CoNLL 2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies*
Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez and Joakim Nivre

Oral Presentations

- 14:30–14:50 *A Latent Variable Model of Synchronous Parsing for Syntactic and Semantic Dependencies*
James Henderson, Paola Merlo, Gabriele Musillo and Ivan Titov
- 14:50–15:10 *Dependency-based Syntactic–Semantic Analysis with PropBank and NomBank*
Richard Johansson and Pierre Nugues
- 15:10–15:30 *A Joint Model for Parsing Syntactic and Semantic Dependencies*
Xavier Lluís and Lluís Màrquez
- 15:30–15:50 *Collective Semantic Role Labelling with Markov Logic*
Sebastian Riedel and Ivan Meza-Ruiz
- 15:50–16:10 *Hybrid Learning of Dependency Structures from Heterogeneous Linguistic Resources*
Yi Zhang, Rui Wang and Hans Uszkoreit
- 16:10–16:20 Closing remarks
- 16:20–16:45 Coffee break

Saturday, August 16, 2008 (continued)

Poster session 16:45–18:00

Parsing Syntactic and Semantic Dependencies with Two Single-Stage Maximum Entropy Models

Hai Zhao and Chunyu Kit

A Combined Memory-Based Semantic Role Labeler of English

Roser Morante, Walter Daelemans and Vincent Van Asch

A Puristic Approach for Joint Dependency Parsing and Semantic Role Labeling

Alexander Volokh and Günter Neumann

Discriminative Learning of Syntactic and Semantic Dependencies

Lu Li, Shixi Fan, Xuan Wang and Xiaolong Wang

Discriminative vs. Generative Approaches in Semantic Role Labeling

Deniz Yuret, Mehmet Ali Yatbaz and Ahmet Engin Ural

A Pipeline Approach for Syntactic and Semantic Dependency Parsing

Yotaro Watanabe, Masakazu Iwatate, Masayuki Asahara and Yuji Matsumoto

Semantic Dependency Parsing using N-best Semantic Role Sequences and Roleset Information

Joo-Young Lee, Han-Cheol Cho and Hae-Chang Rim

A Cascaded Syntactic and Semantic Dependency Parsing System

Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu and Sheng Li

The Integration of Dependency Relation Classification and Semantic Role Labeling Using Bilayer Maximum Entropy Markov Models

Weiwei Sun, Hongzhan Li and Zhifang Sui

Mixing and Blending Syntactic and Semantic Dependencies

Yvonne Samuelsson, Oscar Täckström, Sumithra Velupillai, Johan Eklund, Mark Fishel and Markus Saers

Dependency Tree-based SRL with Proper Pruning and Extensive Feature Engineering

Hongling Wang, Honglin Wang, Guodong Zhou and Qiaoming Zhu

DeSRL: A Linear-Time Semantic Role Labeling System

Massimiliano Ciaramita, Giuseppe Attardi, Felice Dell'Orletta and Mihai Surdeanu

Saturday, August 16, 2008 (continued)

Probabilistic Model for Syntactic and Semantic Dependency Parsing

Enhong Chen, Liu Shi and Dawei Hu

Applying Sentence Simplification to the CoNLL-2008 Shared Task

David Vickrey and Daphne Koller

Sunday, August 17, 2008

Session 1: Language Acquisition I

8:50–9:15 *Picking them up and Figuring them out: Verb-Particle Constructions, Noise and Idiomatcity*

Carlos Ramisch, Aline Villavicencio, Leonardo Moura and Marco Idiart

9:15–9:40 *Fast Mapping in Word Learning: What Probabilities Tell Us*

Afra Alishahi, Afsaneh Fazly and Suzanne Stevenson

9:40–10:05 *Improving Word Segmentation by Simultaneously Learning Phonotactics*

Daniel Blanchard and Jeffrey Heinz

10:05–10:30 *A MDL-based Model of Gender Knowledge Acquisition*

Harmony Marchal, Benoît Lemaire, Maryse Bianco and Philippe Dessus

10:30–11:00 Coffee break

Session 2: Language Acquisition II

11:00–11:25 *Baby SRL: Modeling Early Language Acquisition.*

Michael Connor, Yael Gertner, Cynthia Fisher and Dan Roth

11:25–11:50 *An Incremental Bayesian Model for Learning Syntactic Categories*

Christopher Parisien, Afsaneh Fazly and Suzanne Stevenson

11:50–12:40 Invited talk by Nick Chater

12:40–14:00 Lunch

Sunday, August 17, 2008 (continued)

13:40–14:00 CoNLL Business Meeting

Session 3: Semantic extraction

14:00–14:25 *Fully Unsupervised Graph-Based Discovery of General-Specific Noun Relationships from Web Corpora Frequency Counts*
Gaël Dias, Raycho Mukelov and Guillaume Cleuziou

14:25–14:50 *Acquiring Knowledge from the Web to be used as Selectors for Noun Sense Disambiguation*
Hansen A. Schwartz and Fernando Gomez

14:50–15:15 *Automatic Chinese Catchword Extraction Based on Time Series Analysis*
Han Ren, Donghong Ji, Jing Wan and Lei Han

15:15–15:40 *Easy as ABC? Facilitating Pictorial Communication via Semantically Enhanced Layout*
Andrew B. Goldberg, Xiaojin Zhu, Charles R. Dyer, Mohamed Eldawy and Lijie Heng

15:40–16:00 Coffee break

Session 4: Morphology, MT and Generation

16:00–16:25 *A Nearest-Neighbor Approach to the Automatic Analysis of Ancient Greek Morphology*
John Lee

16:25–16:50 *Context-based Arabic Morphological Analysis for Machine Translation*
Thuy Linh Nguyen and Stephan Vogel

16:50–17:15 *A Tree-to-String Phrase-based Model for Statistical Machine Translation*
Thai Phuong Nguyen, Akira Shimazu, Tu Bao Ho, Minh Le Nguyen and Vinh Van Nguyen

17:15–17:40 *Trainable Speaker-Based Referring Expression Generation*
Giuseppe Di Fabbrizio, Amanda Stent and Srinivas Bangalore

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

Semantic Parsing for High-Precision Semantic Role Labelling

Paola Merlo

Linguistics Department
University of Geneva
5 rue de Candolle
1211 Genève 4 Switzerland
merlo@lettres.unige.ch

Gabriele Musillo

Depts of Linguistics and Computer Science
University of Geneva
5 Rue de Candolle
1211 Genève 4 Switzerland
musillo4@etu.unige.ch

Abstract

In this paper, we report experiments that explore learning of syntactic and semantic representations. First, we extend a state-of-the-art statistical parser to produce a richly annotated tree that identifies and labels nodes with semantic role labels as well as syntactic labels. Secondly, we explore rule-based and learning techniques to extract predicate-argument structures from this enriched output. The learning method is competitive with previous single-system proposals for semantic role labelling, yields the best reported precision, and produces a rich output. In combination with other high recall systems it yields an F-measure of 81%.

1 Introduction

In statistical natural language processing, considerable ingenuity and insight have been devoted to developing models of syntactic information, such as statistical parsers and taggers. Successes in these syntactic tasks have recently paved the way to applying novel statistical learning techniques to levels of semantic representation, such as recovering the logical form of a sentence for information extraction and question-answering applications (Miller et al., 2000; Ge and Mooney, 2005; Zettlemoyer and Collins, 2007; Wong and Mooney, 2007).

In this paper, we also focus our interest on learning semantic information. Differently from other work that has focussed on logical form, however, we explore the problem of recovering the syntactic structure of the sentence, the propositional

argument-structure of its main predicates, and the substantive labels assigned to the arguments in the propositional structure, the semantic roles. This rich output can be useful for information extraction and question-answering, but also for anaphora resolution and other tasks for which the structural information provided by full syntactic parsing is necessary.

The task of semantic role labelling (SRL), as has been defined by previous researchers (Gildea and Jurafsky, 2002), requires collecting all the arguments that together with a verb form a predicate-argument structure. In most previous work, the task has been decomposed into the argument identification and argument labelling subtasks: first the arguments of each specific verb in the sentence are identified by classifying constituents in the sentence as arguments or not arguments. The arguments are then labelled in a second step.

We propose to produce the rich syntactic-semantic output in two steps, which are different from the argument identification and argument labelling subtasks. First, we generate trees that bear both syntactic and semantic annotation, such as those in Figure 1. The parse tree, however, does not explicitly encode information about predicate-argument structure, because it does not explicitly associate each semantic role to the verb that governs it. So, our second step consists in recovering the predicate-argument structure of each verb by gleaning this information in an already richly decorated tree.

There are linguistic and computational reasons to think that we can solve the joint problem of recovering the constituent structure of a sentence and its lexical semantics. From a linguistic point of view, the assumption that syntactic distributions will be predictive of semantic role assignments is based on linking theory (Levin, 1986). Linking theory assumes the existence of a hierarchy of se-

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

mantic roles which are mapped by default on a hierarchy of grammatical functions and syntactic positions, and it attempts to predict the mapping of the underlying semantic component of a predicate’s meaning onto the syntactic structure. For example, Agents are always mapped in syntactically higher positions than Themes. From a computational point of view, if the internal semantics of a predicate determines the syntactic expressions of constituents bearing a semantic role, it is then reasonable to expect that knowledge about semantic roles in a sentence will be informative of its syntactic structure. It follows rather naturally that semantic and syntactic parsing can be integrated into a single complex task.

Our proposal also addresses the problem of semantic role labelling from a slightly different perspective. We identify and label argument nodes first, while parsing, and we group them in a predicate-argument structure in a second step. Our experiments investigate some of the effects that result from organising the task of semantic role labelling in this way, and the usefulness of some novel features defined on syntactic trees.

In the remainder of the paper, we first illustrate the data and the graphical model that formalise the architecture used and its extension for semantic parsing. We then report on two kinds of experiments: we first evaluate the architecture on the joint task of syntactic and semantic parsing and then evaluate the joint approach on the task of semantic role labelling. We conclude with a discussion which highlights the practical and theoretical contribution of this work.

2 The Data

Our experiments on joint syntactic and semantic parsing use data that is produced automatically by merging the Penn Treebank (PTB) with PropBank (PRBK) (Marcus et al., 1993; Palmer et al., 2005), as shown in Figure 1. PropBank encodes propositional information by adding a layer of argument structure annotation to the syntactic structures of the Penn Treebank.¹ Verbal predicates in the Penn Treebank (PTB) receive a label REL and their arguments are annotated with abstract semantic role labels, such as A0, A1, or AA for those complements of the predicative verb that are considered arguments. Those complements of the verb la-

¹We use PRBK data as they appear in the CONLL 2005 shared task.

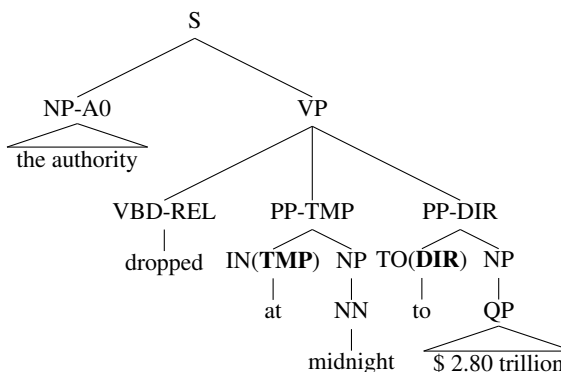


Figure 1: A sample syntactic structure with semantic role labels.

belled with a semantic functional label in the original PTB receive the composite semantic role label AM-*X*, where *X* stands for labels such as LOC, TMP or ADV, for locative, temporal and adverbial modifiers respectively. A tree structure with PropBank labels is shown in Figure 1. (The bold labels are not relevant for the moment and they will be explained later.)

3 The Syntactic and Semantic Parser Architecture

To achieve the complex task of joint syntactic and semantic parsing, we extend a current state-of-the-art statistical parser (Titov and Henderson, 2007) to learn semantic role annotation as well as syntactic structure. The parser uses a form of left-corner parsing strategy to map parse trees to sequences of derivation steps.

We choose this parser because it exhibits the best performance for a single generative parser, and does not impose hard independence assumptions. It is therefore promising for extensions to new tasks. Following (Titov and Henderson, 2007), we describe the original parsing architecture and our modifications to it as a Dynamic Bayesian network. Our description is brief and limited to the few aspects of interest here. For more detail, explanations and experiments see (Titov and Henderson, 2007). A Bayesian network is a directed acyclic graph that illustrates the statistical dependencies between the random variables describing a set of events (Jensen, 2001). Dynamic networks are Bayesian networks applied to unboundedly long sequences. They are an appropriate model for sequences of derivation steps in

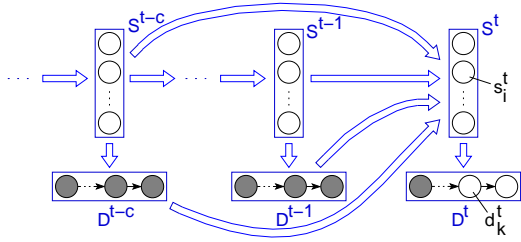


Figure 2: The pattern on connectivity and the latent vectors of variables in an Incremental Bayesian Network.

parsing (Titov and Henderson, 2007).

Figure 2 illustrates visually the main properties that are of relevance for us in this parsing architecture. Let T be a parse tree and D_1, \dots, D_m be the sequence of parsing decisions that has led to the building of this parse tree. Let also each parsing decision be composed of smaller parsing decisions d_1^1, \dots, d_k^1 , and let all these decisions be independent. Then,

$$\begin{aligned} P(T) &= P(D_1, \dots, D_m) \\ &= \prod_t P(D_t | D_1, \dots, D_{t-1}) \quad (1) \\ &= \prod_t \prod_k P(d_k^t | h(t, k)) \end{aligned}$$

where $h(t, k)$ denotes the parse history for sub-decision d_k^t .

The figure represents a small portion of the observed sequence of decisions that constitute the recovery of a parse tree, indicated by the observed states D_i . Specifically, it illustrates the pattern of connectivity for decision d_k^t . As can be seen the relationship between different probabilistic parsing decisions are not Markovian, nor do the decisions influence each other directly. Past decisions can influence the current decision through state vectors of independent latent variables, referred to as S_i . These state vectors encode the probability distributions of features of the history of parsing steps (the features are indicated by s_i^t in Figure 2).

As can be seen from the picture, the pattern of inter-connectivity allows previous non-adjacent states to influence future states. Not all states in the history are relevant, however. The inter-connectivity is defined dynamically based on the topological structure and the labels of the tree that is being developed. This inter-connectivity depends on a notion of structural locality (Henderson, 2003; Musillo and Merlo, 2006).²

²Specifically, the conditioning states are based on the

In order to extend this model to learn decisions concerning a joint syntactic-semantic representation, the semantic information needs to be highlighted in the model in several ways. We modify the network connectivity, and bias the learner.

First, we take advantage of the network’s dynamic connectivity to highlight the portion of the tree that bears semantic information. We augment the nodes that can influence parsing decisions at the current state by explicitly adding the vectors of latent variables related to the most recent child bearing a semantic role label of either type (REL, A0 to A5 or AM-X) to the connectivity of the current decision. These additions yield a model that is sensitive to regularities in structurally defined sequences of nodes bearing semantic role labels, within and across constituents. These extensions enlarge the locality domain over which dependencies between predicates bearing the REL label, arguments bearing an A0-A5 label, and adjuncts bearing an AM-X role can be specified, and capture both linear and hierarchical constraints between predicates, arguments and adjuncts. Enlarging the locality domain this way ensures for instance that the derivation of the role DIR in Figure 1 is not independent of the derivations of the roles TMP, REL (the predicate) and A0.

Second, this version of the Bayesian network tags its sentences internally. Following (Musillo and Merlo, 2005), we split some part-of-speech tags into tags marked with semantic role labels. The semantic role labels attached to a non-terminal directly projected by a preterminal and belonging to a few selected categories (DIR, EXT, LOC, MNR, PRP, CAUS or TMP) are propagated down to the pre-terminal part-of-speech tag of its head.³ This third extension biases the parser to learn the relationship between lexical items, semantic roles and the constituents in which they occur. This technique is illustrated by the bold labels in Figure 1.

We compare this augmented model to a simple baseline parser, that does not present any of the task-specific enhancements described above,

stack configuration of the left-corner parser and the derivation tree built so far. The nodes in the partially built tree and stack configuration that are selected to determine the relevant states are the following: *top*, the node on top of the pushdown stack before the current derivation move; the left-corner ancestor of *top* (that is, the second top-most node on the parser stack); the leftmost child of *top*; and the most recent child of *top*, if any.

³Exploratory data analysis indicates that these tags are the most useful to disambiguate parsing decisions.

	PTB/PRBK 24		
	P	R	F
Baseline	79.6	78.6	79.1
ST	80.5	79.4	79.9
ST+ EC	81.6	80.3	81.0

Table 1: Percentage F-measure (F), recall (R), and precision (P) of our joint syntactic and semantic parser on merged development PTB/PRBK data (section 24). Legend of models: ST=Split Tags; EC=enhanced connectivity.

other than being able to use the complex syntactic-semantic labels. Our augmented model has a total of 613 non-terminals to represent both the PTB and PropBank labels of constituents, instead of the 33 of the original syntactic parser. The 580 newly introduced labels consist of a standard PTB label followed by a set of one or more PropBank semantic role such as PP-AM-TMP or NP-A0-A1. As a result of lowering the six AM- X semantic role labels, 240 new part-of-speech tags were introduced to partition the original tag set which consisted of 45 tags. As already mentioned, argumental labels A0-A5 are specific to a given verb or a given verb sense, thus their distribution is highly variable. To reduce variability, we add the tag-verb pairs licensing these argumental labels to the vocabulary of our model. We reach a total of 4970 tag-word pairs. These pairs include, among others, all the tag-verb pairs occurring at least 10 times in the training data. In this very limited form of lexicalisation, all other words are considered unknown.

4 Parsing Experiments

Our extended joint syntactic and semantic parser was trained on sections 2-21 and validated on section 24 from the merged PTB/PropBank. To evaluate the joint syntactic and semantic parsing task, we compute the standard Parseval measures of labelled recall and precision of constituents, taking into account not only the original PTB labels, but also the newly introduced PropBank labels. This evaluation gives us an indication of how accurately and exhaustively we can recover this richer set of syntactic and semantic labels. The results, computed on the development data set from section 24 of the PTB with added PropBank annotation, are shown in Table 1. As the table indicates, both the enhancements based on semantic roles yield an im-

provement on the baseline.

This task enables us to compare, albeit indirectly, our integrated method to other methods where semantic role labels are learnt separately from syntactic structure. (Musillo and Merlo, 2006) report results of a merging technique where the output of the semantic role annotation produced by the best semantic role labellers in the 2005 CoNLL shared task is merged with the output of Charniak’s parser. Results range between 82.7% and 83.4% F-measure. Our integrated method almost reaches this level of performance.

The performance of the parser on the syntactic labels only (note reported in Table 1) is slightly degraded in comparison to the original syntax-only architecture (Henderson, 2003), which reported an F-measure of 89.1% since we reach 88.4% F-measure for the best syntactic-semantic model (last line of Table 1). This level of performance is still comparable to other syntactic parsers often used for extraction of semantic role features (88.2% F-measure) (Collins, 1999).

These results indicate that the extended parser is able to recover both syntactic and semantic labels in a fully connected parse tree. While it is true that the full fine-grained interpretation of the semantic label is verb-specific, the PropBank labels (A0,A1, etc) do respect some general trends. A0 labels are assigned to the most agentive of the arguments, while A1 labels tend to be assigned to arguments bearing a Theme role, and A2, A3, A4 and A5 labels are assigned to indirect object roles, while all the AM- X labels tend to be assigned to adjuncts. The fact that the parser learns these labels without explicit annotation of the link between the arguments and the predicate to which they are assigned, but based on the smoothed representation of the derivation of the parse tree and only very limited lexicalisation, appears to confirm linking theory, which assumes a correlation between the syntactic configuration of a sentence and the lexical semantic labels.

We need to show now that the quality of the output produced by the joint syntactic and semantic parsing is such that it can be used to perform other tasks where semantic role information is crucial. The most directly related task is semantic role labelling (SRL) as defined in the shared task of CoNLL 2005.

5 Extraction of Predicate-Argument Structures

Although there is reason to think that the good performance reported in the previous section is due to implicit learning of the relationship of the syntactic representation and the semantic role assignments, the output produced by the parser does not explicitly encode the predicate-argument structures. Collecting these associations is required to solve the semantic role labelling task as usually defined. We experimented with two methods: a simple rule-based method and a more complex learning method.

5.1 The rule-based method

The rule-based extraction method is the natural second step to solve the complete semantic role labelling task, after we identify and label semantic roles while parsing. Since in our proposal, we solve most of the problem in the first step, then we should be able to collect the predicate-argument pairs by simple, deterministic rules. The simplicity of the method also provides a useful comparison for more complex learning methods, which can be justified only if they perform better than simple rule-based predicate-argument extraction.

Our rule-based method automatically compiles finite-state automata defining the paths that connect the first node dominating a predicate to its semantic roles from parse trees enriched with semantic role labels.⁴ Such paths can then be used to traverse parse trees returned by the parsing model and collect argument structures. More specifically, a sample of sentences are randomly selected from the training section of the PTB/PRBK. For each predicate, then, all the arguments left and right of the predicate and all the adjuncts left and right respectively are collected and filtered by simple global constraints, thereby guaranteeing that only one type of obligatory argument label (A0 to A5) is assigned in each proposition.

When evaluated on gold data, this rule-based extraction method reaches 94.9% precision, 96.9% recall, for an F-measure of 95.9%. These results provide an upper bound as well as indicating that, while not perfect, the simple extraction rules reach a very good level of correctness if the input from the first step, syntactic and semantic parsing, is correct. The performance is much lower when

parses are not entirely correct, and semantic role labels are missing, as indicated by the results of 72.9% precision, 66.7% (F-measure 69.7%), obtained when using the best automatic parse tree. The fact that performance depends on the quality of the output of the first step, indicates that the extraction rules are sensitive to errors in the parse trees, as well as errors in the labelling. This indicates that a learning method might be more adapted to recover from these mistakes.

5.2 The SVM learning method

In a different approach to extract predicate argument structures from the parsing output, the second step learns to associate the right verb to each semantically annotated node (*srn*) in the tree produced in the first step. Each individual (*verb*, *srn*) pair in the tree is either a positive example (the *srn* is a member of the verb's argument structure) or a negative example (the argument either should not have been labelled as an argument or it is not associated to the verb). The training examples are produced by parsing section 2-21 of the merged PTB/PRBK data with the joint syntactic-semantic parser and producing the training examples by comparison with the CONLL 2005 gold propositions. There are approximately 800'000 training examples in total. These examples are used by an SVM classifier (Joachims, 1999).⁵ Once the predicate-argument structures are built, they are evaluated with the CONLL 2005 shared task criteria.

5.3 The learning features

The features used for the extraction of the predicate-argument structure reflect the syntactic properties that are useful to identify the arguments of a given verb. We use syntactic and semantic node label, the path between the verb and the argument, and the part-of-speech tag of the verb, which provides useful information about the tense of the verb. We also use novel features that encode *minimality conditions* and *locality constraints* (Rizzi, 1990). Minimality is a typical property of natural languages that is attested in several domains. In recovering predicate-argument structures, minimality guarantees that the arguments are related to the closest verb in a predicate domain, which is not always the verb to which they are connected by the

⁴It uses VanNoord's finite-state-toolkit <http://www.let.rug.nl/vannoord/Fsa/>.

⁵We use a radial basis function kernel, where parameters c and γ were determined by a grid search on a small subset of 2000 training examples. They are set at $c=8$ and $\gamma = 0.03125$.

shortest path. For example, the subject of an embedded clause can be closer to the verb of the main clause than to the predicate to which it should be attached. Minimality is encoded as a binary feature that indicates whether a verb w intervenes between the verb v and the candidate argument srn . Minimality is defined both in terms of linear precedence (indicated below as \prec) and of dominance within the same VP group. A VP group is a stack of VPs covering the same compound verb group, such as $[_{VP} \textit{should} [_{VP} \textit{have} [_{VP} [_{V} \textit{come}]]]]$. Formal definitions are given below:

$$\textit{minimal}(v, srn, w) =_{df}$$

$$\begin{cases} \textit{false} & \text{if } (v \prec w \prec srn \text{ or } srn \prec w \prec v) \text{ and} \\ & \text{VPG-dominates}(v, srn, w) \\ \textit{true} & \text{otherwise} \end{cases}$$

$$\textit{VPG-dominates}(v, srn, w) =_{df}$$

$$\begin{cases} \textit{true} & \text{if VP} \in \text{path}(v, srn) \text{ and} \\ & \text{VP} \in \text{VP-group directly dominating } w \\ \textit{false} & \text{otherwise} \end{cases}$$

In addition to the minimality conditions, which resolve ambiguity when two predicates compete to govern an argument, we use locality constraints to capture distinct local relationships between a verb and the syntactic position occupied by a candidate argument. In particular, we distinguish between internal arguments occupying a position dominated by a VP node, external arguments occupying a position dominated by an S node, and extracted arguments occupying a position dominated by an SBAR node. To approximate such structural distinctions, we introduce two binary features indicating, respectively, whether there is a node labelled S or SBAR on the path connecting the verb and the candidate argument.

6 Results and Discussion

Table 2 illustrates our results on semantic role labelling. Notice how much more precise the learning method is than the rule-based method, when the minimality constraint is added. The second and third line indicate that this contribution is mostly due to the minimality feature. The fifth and sixth line however illustrate that these features together are more useful than the widely used feature *path*. Recall however, suffers in the learnt method. Overall, the learnt method is better than a rule-based method only if path and either minimality or locality constraints are added, thus suggesting that

	Prec	Rec	F
Learning all features	87.4	63.6	73.7
Learning all –min	75.4	66.2	70.5
Learning all –loc	87.4	63.6	73.6
Rule-based	72.9	66.7	69.7
Learning all –path	80.6	60.9	69.4
Learning all –min –loc	74.3	63.8	68.6
Baseline	57.4	53.9	55.6

Table 2: Results on the development section (24), rule-based, and learning, (with all features, and without path, minimality and locality constraints) compared to a closest verb baseline.

the choice of features is crucial to reach a level of performance that justifies the added complexity of a learning method. Both methods are much better than a baseline that attaches each role to a verb by the shortest path.⁶ Notice that both these approaches are not lexicalised, they apply to all verbs. Learning experiments where the actual verbs were used showed a little degradation as well as a very considerable increase in training times (precision: 87.0%; recall: 61.0%; F: 71.7%).⁷

Some comments are in order to compare properly our best results – the learning method with all features – to other methods. Most of the best performing SRL systems are ensemble learners or rerankers, or they use external sources of information such as the PropBank frames files. While these techniques are effective to improve classification accuracy, we might want to compare the single systems, thus teasing apart the contribution of the features and the model from the contribution of the ensemble technique. Table 3 reports the single systems’ performance on the test set. These results seem to indicate that methods like ours, based on a first step of PropBank parsing, are comparable to other methods when learning regimes are factored out, contrary to pessimistic conclusions in previous work (Yi and Palmer, 2005). (Yi and Palmer, 2005) share the motivation of our work. They observe that the distributions of semantic la-

⁶In case of tie, the following verb is chosen for an A0 label and the preceding verb is chosen for all the other labels.

⁷We should notice that all these models encode the feature path as syntactic path, because in exploratory data analysis we found that this feature performed quite a bit better than path encoded taking into account the semantic roles assigned to the nodes on the path. Concerning the learning model, we notice that a simpler, and much faster to train, linear SVM classifier performs almost as well as the more complex RBF classifier. It is therefore preferable if speed is important.

Model	CONLL 23			Comments
	P	R	F	
(Surdeanu and Turmo, 2005)	80.3	73.0	76.5	Propbank frames to filter output, boosting
(Liu et al., 2005)	80.5	72.8	76.4	Single system + simple post-processing
(Moschitti et al., 2005)	76.6	75.2	75.9	Specialised kernels for each kind of role
This paper	87.6	65.8	75.1	Single system and model, locality features
(Ozgenicil and McCracken, 2005)	74.7	74.2	74.4	Simple system, no external knowledge
(Johansson and Nugues, 2005)	75.5	73.2	74.3	Uses only 3 sections for training

Table 3: Final Semantic Role Labelling results on test section 23 of Propbank as encoded in the CONLL shared task for those CONLL 2005 participants not using ensemble learning or external resources.

bels could potentially interact with the distributions of syntactic labels and redefine the boundaries of constituents, thus yielding trees that reflect generalisations over both these sources of information. They also attempt to assign SRL while parsing, by merging only the first two steps of the standard pipeline architecture, pruning and argument identification. Their parser outputs a binary argument-nonargument distinction. The actual fine-grained labelling is performed, as in other methods, by an ensemble classifier. Results are not among the best and Yi and Palmer conclude that PropBank parsing is too difficult and suffers from differences between chunk annotation and tree structure. We think instead that the method is promising, as shown by the results reported here, once the different factors that affect performance are teased apart.

Some qualitative observations on the errors are useful. On the one hand, as can be noticed in Table 3, our learning method yields the best precision, but often the worse recall and it has the most extreme difference between these two scores.⁸ This is very likely to be a consequence of the method. Since the assignment of the semantic role labels proper is performed during parsing, the number of nodes that require a semantic role is only 20% of the total. Therefore the parser develops a bias against assigning these roles in general, and recall suffers.⁹ On the other hand, precision is very good, thanks to the rich context in which the roles are assigned.

This property of our method suggests that combining our results with those of other existing se-

⁸This observation applies also in a comparison to the other systems that participated in the CONLL shared task.

⁹The SVM classifier, on the other hand, exceeds 94% in accuracy and its F measures are situated around 87–88% depending on the feature sets.

mantic role labellers might be beneficial, since the errors it performs are quite different. We tested this hypothesis by combining our outputs, which are the most precise, with the outputs of the system that reported the best recall (Haghighi et al., 2005). The combination, performed on sections 24 and 23, gives priority to our system when it outputs a non-null label (because of its high precision) and uses the other system’s label when our system outputs a null label. This combination produces a result of 79.0% precision, 80.4% recall, and 79.7% F-measure for section 24, and 80.5% precision, 81.4% recall, and 81.0% F-measure for section 23. We conclude that the combination is indeed able to exploit the positive aspects of both approaches, as the F-measure of the combined result is better than each individual result. It is also the best compared to the other systems of the CoNLL shared task. Comparatively, we find that applying the same combination technique to the output of the system by (Haghighi et al., 2005) with the output of the best system in the CoNLL 2005 shared task (Punyakankok et al., 2005) yields combined outputs that are not as good as the better of the two systems (P:76.3%; R:78.6%; F:77.4% for section 24; P:78.5%; R:80.0%; F:79.3% for section 23). This result confirms our initial hypothesis, that combination of systems with different performance characteristics yields greater improvement.

Another direct consequence of assigning roles in a rich context is that in collecting arguments for a given verb we hardly need to verify global constraints. Differently from previous work that had found that global coherence constraints considerably improved performance (Punyakankok et al., 2005), using global filtering constraints showed no improvement in our learning model. Thus, these results confirm the observations that a verb does

not assign its semantic roles independently of each other (Haghighi et al., 2005). Our method too can be seen as a way of formulating the SRL problem in a way that is not simply classification of each instance independently. Because identification of arguments and their labelling is done while parsing, the parsing history, both syntactic and semantic, is taken into account in identifying and labelling an argument. Semantic role labelling is integrated in structured sequence prediction. Further integration of semantic role labelling in structured probabilistic models related to the one described here has recently been shown to result in accurate synchronous parsers that derive both syntactic and semantic dependency representations (Henderson et al., 2008).

7 Conclusion

Overall our experiments indicate that an integrated approach to identification and labelling followed by predicate-argument recovery can solve the problem of semantic role labelling at a level of performance comparable to other approaches, as well as yielding a richly decorated syntactic-semantic parse tree. The high precision of our method yields very good results in combination with other high-recall systems. Its shortcomings indicates that future work lies in improving recall.

Acknowledgments

We thank the Swiss NSF for supporting this research under grant number 101411-105286/1, James Henderson for sharing the SSN software, and Xavier Carreras for providing the CoNLL-2005 data. Part of this work was completed while the second author was visiting MIT/CSAIL, hosted by Prof. Michael Collins.

References

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

Ge, Ruiyang and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Procs of CONLL-05*, Ann Arbor, Michigan.

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

Haghighi, Aria, Kristina Toutanova, and Christopher Manning. 2005. A joint model for semantic role labeling. In *Procs of CoNLL-2005*, pages 173–176, Ann Arbor, Michigan.

Henderson, Jamie. 2003. Inducing history representations for broad-coverage statistical parsing. In *Procs of NAACL-HLT'03*, pages 103–110, Edmonton, Canada.

Henderson, James, Paola Merlo, Gabriele Musillo and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Procs of CoNLL'08 Shared Task*, Manchester, UK.

Jensen, Finn V. 2001. *Bayesian networks and decision graphs*. Springer Verlag.

Joachims, Thorsten. 1999. Making large-scale svm learning practical. In Schlkopf, B., C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Johansson, Richard and Pierre Nugues. 2005. Sparse bayesian classification of predicate arguments. In *Procs of CoNLL-2005*, pages 177–180, Ann Arbor, Michigan.

Levin, Lori. 1986. *Operations on lexical form: unaccusative rules in Germanic languages*. Ph.D. thesis, Massachusetts Institute of Technology.

Liu, Ting, Wanxiang Che, Sheng Li, Yuxuan Hu, and Huaijun Liu. 2005. Semantic role labeling system using maximum entropy classifier. In *Procs of CoNLL-2005*, pages 189–192, Ann Arbor, Michigan.

Marcus, Mitch, Beatrice Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

Miller, S., H. Fox, L. Ramshaw, and R. Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Procs of NAACL 2000*.

Moschitti, Alessandro, Ana-Maria Giuglea, Bonaventura Coppola, and Roberto Basili. 2005. Hierarchical semantic role labeling. In *Procs of CoNLL-2005*, pages 201–204, Ann Arbor, Michigan.

Musillo, Gabriele and Paola Merlo. 2005. Lexical and structural biases for function parsing. In *Procs of IWPT'05*, pages 83–92, Vancouver, British Columbia, October.

Musillo, Gabriele and Paola Merlo. 2006. Accurate semantic parsing of the proposition bank. In *Procs of NAACL'06*, New York, NY.

Ozgenicil, Necati Ercan and Nancy McCracken. 2005. Semantic role labeling using libSVM. In *Procs of CoNLL-2005*, pages 205–208, Ann Arbor, Michigan, June.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31:71–105.

Punyakanok, Vasin, Peter Koomen, Dan Roth, and Wen tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Procs of CoNLL-2005*, Ann Arbor, MI USA.

Rizzi, Luigi. 1990. *Relativized minimality*. MIT Press, Cambridge, MA.

Surdeanu, Mihai and Jordi Turmo. 2005. Semantic role labeling using complete syntactic analysis. In *Procs of CoNLL'05*, Ann Arbor, Michigan.

Titov, Ivan and James Henderson. 2007. Constituent parsing with Incremental Sigmoid Belief Networks. In *Procs of ACL'07*, pages 632–639, Prague, Czech Republic.

Wong, Yuk Wah and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Procs of ACL'07*, pages 960–967, Prague, Czech Republic.

Yi, Szu-ting and Martha Palmer. 2005. The integration of semantic parsing and semantic role labelling. In *Procs of CoNLL'05*, Ann Arbor, Michigan.

Zettlemoyer, Luke and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Procs of EMNLP-CoNLL'07*, pages 678–687.

TAG, Dynamic Programming, and the Perceptron for Efficient, Feature-rich Parsing

Xavier Carreras Michael Collins Terry Koo
MIT CSAIL, Cambridge, MA 02139, USA
{carreras, mcollins, maestro}@csail.mit.edu

Abstract

We describe a parsing approach that makes use of the perceptron algorithm, in conjunction with dynamic programming methods, to recover full constituent-based parse trees. The formalism allows a rich set of parse-tree features, including PCFG-based features, bigram and trigram dependency features, and surface features. A severe challenge in applying such an approach to full syntactic parsing is the efficiency of the parsing algorithms involved. We show that efficient training is feasible, using a Tree Adjoining Grammar (TAG) based parsing formalism. A lower-order dependency parsing model is used to restrict the search space of the full model, thereby making it efficient. Experiments on the Penn WSJ treebank show that the model achieves state-of-the-art performance, for both constituent and dependency accuracy.

1 Introduction

In global linear models (GLMs) for structured prediction, (e.g., (Johnson et al., 1999; Lafferty et al., 2001; Collins, 2002; Altun et al., 2003; Taskar et al., 2004)), the optimal label y^* for an input \mathbf{x} is

$$y^* = \arg \max_{y \in \mathcal{Y}(\mathbf{x})} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, y) \quad (1)$$

where $\mathcal{Y}(\mathbf{x})$ is the set of possible labels for the input \mathbf{x} ; $\mathbf{f}(\mathbf{x}, y) \in \mathbb{R}^d$ is a feature vector that represents the pair (\mathbf{x}, y) ; and \mathbf{w} is a parameter vector. This paper describes a GLM for natural language parsing, trained using the averaged perceptron. The parser we describe recovers full syntactic representations, similar to those derived by a probabilistic context-free grammar (PCFG). A key motivation for the use of GLMs in parsing is that they allow a great deal of flexibility in the features which can be included in the definition of $\mathbf{f}(\mathbf{x}, y)$.

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

A critical problem when training a GLM for parsing is the computational complexity of the inference problem. The averaged perceptron requires the training set to be repeatedly decoded under the model; under even a simple PCFG representation, finding the $\arg \max$ in Eq. 1 requires $O(n^3G)$ time, where n is the length of the sentence, and G is a grammar constant. The average sentence length in the data set we use (the Penn WSJ treebank) is over 23 words; the grammar constant G can easily take a value of 1000 or greater. These factors make exact inference algorithms virtually intractable for training or decoding GLMs for full syntactic parsing.

As a result, in spite of the potential advantages of these methods, there has been very little previous work on applying GLMs for full parsing without the use of fairly severe restrictions or approximations. For example, the model in (Taskar et al., 2004) is trained on only sentences of 15 words or less; reranking models (Collins, 2000; Charniak and Johnson, 2005) restrict $\mathcal{Y}(\mathbf{x})$ to be a small set of parses from a first-pass parser; see section 1.1 for discussion of other related work.

The following ideas are central to our approach:

(1) A TAG-based, splittable grammar. We describe a novel, TAG-based parsing formalism that allows full constituent-based trees to be recovered. A driving motivation for our approach comes from the flexibility of the feature-vector representations $\mathbf{f}(\mathbf{x}, y)$ that can be used in the model. The formalism that we describe allows the incorporation of: (1) basic PCFG-style features; (2) the use of features that are sensitive to *bigram* dependencies between pairs of words; and (3) features that are sensitive to *trigram* dependencies. Any of these feature types can be combined with *surface features* of the sentence \mathbf{x} , in a similar way

to the use of surface features in conditional random fields (Lafferty et al., 2001). Crucially, in spite of these relatively rich representations, the formalism can be parsed efficiently (in $O(n^4G)$ time) using dynamic-programming algorithms described by Eisner (2000) (unlike many other TAG-related approaches, our formalism is “splittable” in the sense described by Eisner, leading to more efficient parsing algorithms).

(2) Use of a lower-order model for pruning.

The $O(n^4G)$ running time of the TAG parser is still too expensive for efficient training with the perceptron. We describe a method that leverages a simple, first-order dependency parser to restrict the search space of the TAG parser in training and testing. The lower-order parser runs in $O(n^3H)$ time where $H \ll G$; experiments show that it is remarkably effective in pruning the search space of the full TAG parser.

Experiments on the Penn WSJ treebank show that the model recovers constituent structures with higher accuracy than the approaches of (Charniak, 2000; Collins, 2000; Petrov and Klein, 2007), and with a similar level of performance to the reranking parser of (Charniak and Johnson, 2005). The model also recovers dependencies with significantly higher accuracy than state-of-the-art dependency parsers such as (Koo et al., 2008; McDonald and Pereira, 2006).

1.1 Related Work

Previous work has made use of various restrictions or approximations that allow efficient training of GLMs for parsing. This section describes the relationship between our work and this previous work.

In *reranking* approaches, a first-pass parser is used to enumerate a small set of candidate parses for an input sentence; the reranking model, which is a GLM, is used to select between these parses (e.g., (Ratnaparkhi et al., 1994; Johnson et al., 1999; Collins, 2000; Charniak and Johnson, 2005)). A crucial advantage of our approach is that it considers a very large set of alternatives in $\mathcal{Y}(\mathbf{x})$, and can thereby avoid search errors that may be made in the first-pass parser.¹

Another approach that allows efficient training of GLMs is to use *simpler syntactic representations*, in particular dependency structures (McDon-

¹Some features used within reranking approaches may be difficult to incorporate within dynamic programming, but it is nevertheless useful to make use of GLMs in the dynamic-programming stage of parsing. Our parser could, of course, be used as the first-stage parser in a reranking approach.

ald et al., 2005). Dependency parsing can be implemented in $O(n^3)$ time using the algorithms of Eisner (2000). In this case there is no grammar constant, and parsing is therefore efficient. A disadvantage of these approaches is that they do not recover full, constituent-based syntactic structures; the increased linguistic detail in full syntactic structures may be useful in NLP applications, or may improve dependency parsing accuracy, as is the case in our experiments.²

There has been some previous work on GLM approaches for full syntactic parsing that make use of dynamic programming. Taskar et al. (2004) describe a max-margin approach; however, in this work training sentences were limited to be of 15 words or less. Clark and Curran (2004) describe a log-linear GLM for CCG parsing, trained on the Penn treebank. This method makes use of parallelization across an 18 node cluster, together with up to 25GB of memory used for storage of dynamic programming structures for training data. Clark and Curran (2007) describe a perceptron-based approach for CCG parsing which is considerably more efficient, and makes use of a super-tagging model to prune the search space of the full parsing model. Recent work (Petrov et al., 2007; Finkel et al., 2008) describes log-linear GLMs applied to PCFG representations, but does not make use of dependency features.

2 The TAG-Based Parsing Model

2.1 Derivations

This section describes the idea of *derivations* in our parsing formalism. As in context-free grammars or TAGs, a derivation in our approach is a data structure that specifies the sequence of operations used in combining basic (elementary) structures in a grammar, to form a full parse tree. The parsing formalism we use is related to the tree adjoining grammar (TAG) formalisms described in (Chiang, 2003; Shen and Joshi, 2005). However, an important difference of our work from this previous work is that our formalism is defined to be “splittable”, allowing use of the efficient parsing algorithms of Eisner (2000).

A derivation in our model is a pair $\langle E, D \rangle$ where E is a set of *spines*, and D is a set of *dependencies*

²Note however that the lower-order parser that we use to restrict the search space of the TAG-based parser is based on the work of McDonald et al. (2005). See also (Sagae et al., 2007) for a method that uses a dependency parser to restrict the search space of a more complex HPSG parser.

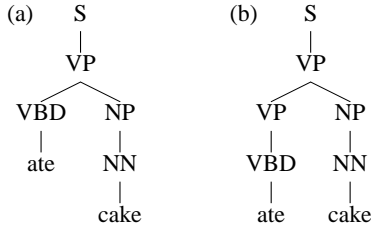
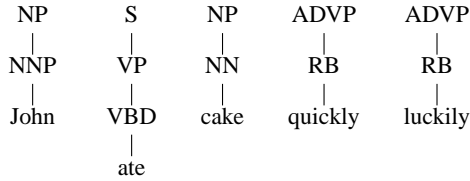


Figure 1: Two example trees.

specifying how the spines are combined to form a parse tree. The spines are similar to elementary trees in TAG. Some examples are as follows:



These structures do not have substitution nodes, as is common in TAGs.³ Instead, the spines consist of a lexical anchor together with a series of unary projections, which usually correspond to different X-bar levels associated with the anchor.

The operations used to combine spines are similar to the TAG operations of adjunction and sister adjunction. We will call these operations *regular adjunction* (r-adjunction) and *sister adjunction* (s-adjunction). As one example, the *cake* spine shown above can be s-adjoined into the VP node of the *ate* spine, to form the tree shown in figure 1(a). In contrast, if we use the r-adjunction operation to adjoin the *cake* tree into the VP node, we get a different structure, which has an additional VP level created by the r-adjunction operation: the resulting tree is shown in figure 1(b). The r-adjunction operation is similar to the usual adjunction operation in TAGs, but has some differences that allow our grammars to be splittable; see section 2.3 for more discussion.

We now give formal definitions of the sets E and D . Take \mathbf{x} to be a sentence consisting of $n + 1$ words, $x_0 \dots x_n$, where x_0 is a special *root* symbol, which we will denote as $*$. A derivation for the input sentence \mathbf{x} consists of a pair $\langle E, D \rangle$, where:

- E is a set of $(n + 1)$ tuples of the form $\langle i, \eta \rangle$, where $i \in \{0 \dots n\}$ is an index of a word in the sentence, and η is the spine associated with the word x_i . The set E specifies one spine for each of the $(n + 1)$ words in the sentence. Where it is

³It would be straightforward to extend the approach to include substitution nodes, and a substitution operation.

clear from context, we will use η_i to refer to the spine in E corresponding to the i 'th word.

- D is a set of n dependencies. Each dependency is a tuple $\langle h, m, l \rangle$. Here h is the index of the *head-word* of the dependency, corresponding to the spine η_h which contains a node that is being adjoined into. m is the index of the *modifier-word* of the dependency, corresponding to the spine η_m which is being adjoined into η_h . l is a *label*.

The label l is a tuple $\langle \text{POS}, A, \eta_h, \eta_m, L \rangle$. η_h and η_m are the head and modifier spines that are being combined. POS specifies which node in η_h is being adjoined into. A is a binary flag specifying whether the combination operation being used is s-adjunction or r-adjunction. L is a binary flag specifying whether or not any “previous” modifier has been r-adjoined into the position POS in η_h . By a previous modifier, we mean a modifier m' that was adjoined from the same direction as m (i.e., such that $h < m' < m$ or $m < m' < h$).

It would be sufficient to define l to be the pair $\langle \text{POS}, A \rangle$ —the inclusion of η_h , η_m and L adds redundant information that can be recovered from the set E , and other dependencies in D —but it will be convenient to include this information in the label. In particular, it is important that given this definition of l , it is possible to define a function $\text{GRM}(l)$ that maps a label l to a triple of non-terminals that represents the grammatical relation between m and h in the dependency structure. For example, in the tree shown in figure 1(a), the grammatical relation between *cake* and *ate* is the triple $\text{GRM}(l) = \langle \text{VP VBD NP} \rangle$. In the tree shown in figure 1(b), the grammatical relation between *cake* and *ate* is the triple $\text{GRM}(l) = \langle \text{VP VP NP} \rangle$.

The conditions under which a pair $\langle E, D \rangle$ forms a valid derivation for a sentence \mathbf{x} are similar to those in conventional LTAGs. Each $\langle i, \eta \rangle \in E$ must be such that η is an elementary tree whose anchor is the word x_i . The dependencies D must form a directed, projective tree spanning words $0 \dots n$, with $*$ at the root of this tree, as is also the case in previous work on discriminative approaches to dependency parsing (McDonald et al., 2005). We allow any modifier tree η_m to adjoin into any position in any head tree η_h , but the dependencies D must nevertheless be coherent—for example they must be consistent with the spines in E , and they must be nested correctly.⁴ We will al-

⁴For example, closer modifiers to a particular head must adjoin in at the same or a lower spine position than modifiers

low multiple modifier spines to s-adjoin or r-adjoin into the same node in a head spine; see section 2.3 for more details.

2.2 A Global Linear Model

The model used for parsing with this approach is a global linear model. For a given sentence \mathbf{x} , we define $\mathcal{Y}(\mathbf{x})$ to be the set of valid derivations for \mathbf{x} , where each $y \in \mathcal{Y}(\mathbf{x})$ is a pair $\langle E, D \rangle$ as described in the previous section. A function \mathbf{f} maps (\mathbf{x}, y) pairs to feature-vectors $\mathbf{f}(\mathbf{x}, y) \in \mathbb{R}^d$. The parameter vector \mathbf{w} is also a vector in \mathbb{R}^d . Given these definitions, the optimal derivation for an input sentence \mathbf{x} is $y^* = \arg \max_{y \in \mathcal{Y}(\mathbf{x})} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, y)$.

We now come to how the feature-vector $\mathbf{f}(\mathbf{x}, y)$ is defined in our approach. A simple “first-order” model would define

$$\mathbf{f}(\mathbf{x}, y) = \sum_{\langle i, \eta \rangle \in E(y)} \mathbf{e}(\mathbf{x}, \langle i, \eta \rangle) + \sum_{\langle h, m, l \rangle \in D(y)} \mathbf{d}(\mathbf{x}, \langle h, m, l \rangle) \quad (2)$$

Here we use $E(y)$ and $D(y)$ to respectively refer to the set of spines and dependencies in y . The function \mathbf{e} maps a sentence \mathbf{x} paired with a spine $\langle i, \eta \rangle$ to a feature vector. The function \mathbf{d} maps dependencies within y to feature vectors. This decomposition is similar to the first-order model of McDonald et al. (2005), but with the addition of the \mathbf{e} features.

We will extend our model to include higher-order features, in particular features based on *sibling* dependencies (McDonald and Pereira, 2006), and *grandparent* dependencies, as in (Carreras, 2007). If $y = \langle E, D \rangle$ is a derivation, then:

- $S(y)$ is a set of sibling dependencies. Each sibling dependency is a tuple $\langle h, m, l, s \rangle$. For each $\langle h, m, l, s \rangle \in S$ the tuple $\langle h, m, l \rangle$ is an element of D ; there is one member of S for each member of D . The index s is the index of the word that was adjoined to the spine for h immediately before m (or the NULL symbol if no previous adjunction has taken place).

- $G(y)$ is a set of grandparent dependencies of type 1. Each type 1 grandparent dependency is a tuple $\langle h, m, l, g \rangle$. There is one member of G for every member of D . The additional information, the index g , is the index of the word that is the first modifier to the *right* of the spine for m .

that are further from the head.

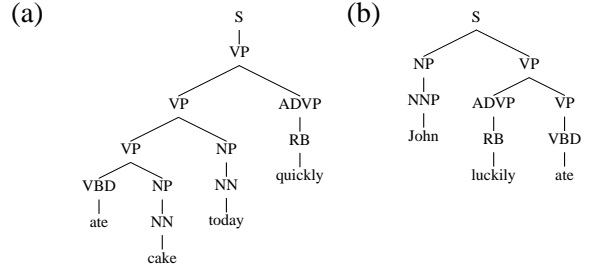


Figure 2: Two Example Trees

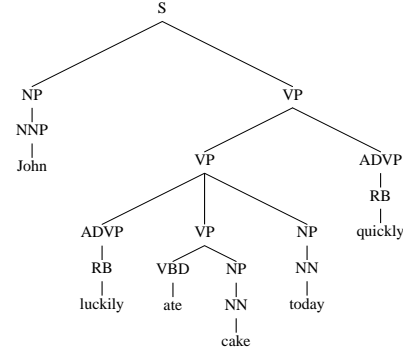


Figure 3: An example tree, formed by a combination of the two structures in figure 2.

- $Q(y)$ is an additional set of grandparent dependencies, of type 2. Each of these dependencies is a tuple $\langle h, m, l, q \rangle$. Again, there is one member of Q for every member of D . The additional information, the index q , is the index of the word that is the first modifier to the *left* of the spine for m .

The feature-vector definition then becomes:

$$\mathbf{f}(\mathbf{x}, y) = \sum_{\langle i, \eta \rangle \in E(y)} \mathbf{e}(\mathbf{x}, \langle i, \eta \rangle) + \sum_{\langle h, m, l \rangle \in D(y)} \mathbf{d}(\mathbf{x}, \langle h, m, l \rangle) + \sum_{\langle h, m, l, s \rangle \in S(y)} \mathbf{s}(\mathbf{x}, \langle h, m, l, s \rangle) + \sum_{\langle h, m, l, g \rangle \in G(y)} \mathbf{g}(\mathbf{x}, \langle h, m, l, g \rangle) + \sum_{\langle h, m, l, q \rangle \in Q(y)} \mathbf{q}(\mathbf{x}, \langle h, m, l, q \rangle) \quad (3)$$

where \mathbf{s} , \mathbf{g} and \mathbf{q} are feature vectors corresponding to the new, higher-order elements.⁵

2.3 Recovering Parse Trees from Derivations

As in TAG approaches, there is a mapping from derivations $\langle E, D \rangle$ to parse trees (i.e., the type of trees generated by a context-free grammar). In our case, we map a spine and its dependencies to a constituent structure by first handling the dependen-

⁵We also added constituent-boundary features to the model, which is a simple change that led to small improvements on validation data; for brevity we omit the details.

cies on each side separately and then combining the left and right sides.

First, it is straightforward to build the constituent structure resulting from multiple adjunctions on the same side of a spine. As one example, the structure in figure 2(a) is formed by first s-adjointing the spine with anchor *cake* into the VP node of the spine for *ate*, then r-adjointing spines anchored by *today* and *quickly* into the same node, where all three modifier words are to the right of the head word. Notice that each r-adjunction operation creates a new VP level in the tree, whereas s-adjunctions do not create a new level. Now consider a tree formed by first r-adjointing a spine for *luckily* into the VP node for *ate*, followed by s-adjointing the spine for *John* into the S node, in both cases where the modifiers are to the left of the head. In this case the structure that would be formed is shown in figure 2(b).

Next, consider combining the left and right structures of a spine. The main issue is how to handle multiple r-adjunctions or s-adjunctions on both sides of a node in a spine, because our derivations do not specify how adjunctions from different sides embed with each other. In our approach, the combination operation preserves the height of the different modifiers from the left and right directions. To illustrate this, figure 3 shows the result of combining the two structures in figure 2. The combination of the left and right modifier structures has led to flat structures, for example the rule $VP \rightarrow ADVP VP NP$ in the above tree.

Note that our r-adjunction operation is different from the usual adjunction operation in TAGs, in that “wrapping” adjunctions are not possible, and r-adjunctions from the left and right directions are independent from each other; because of this our grammars are splittable.

3 Parsing Algorithms

3.1 Use of Eisner’s Algorithms

This section describes the algorithm for finding $y^* = \arg \max_{y \in \mathcal{Y}(\mathbf{x})} \mathbf{w} \cdot \mathbf{f}(\mathbf{x}, y)$ where $\mathbf{f}(\mathbf{x}, y)$ is defined through either the first-order model (Eq. 2) or the second-order model (Eq. 3).

For the first-order model, the methods described in (Eisner, 2000) can be used for the parsing algorithm. In Eisner’s algorithms for dependency parsing each word in the input has left and right finite-state (weighted) automata, which generate the left and right modifiers of the word in question. We

make use of this idea of automata, and also make direct use of the method described in section 4.2 of (Eisner, 2000) that allows a set of possible senses for each word in the input string. In our use of the algorithm, each possible sense for a word corresponds to a different possible spine that can be associated with that word. The left and right automata are used to keep track of the last position in the spine that was adjoined into on the left/right of the head respectively. We can make use of separate left and right automata—i.e., the grammar is splittable—because left and right modifiers are adjoined independently of each other in the tree. The extension of Eisner’s algorithm to the second-order model is similar to the algorithm described in (Carreras, 2007), but again with explicit use of word senses and left/right automata. The resulting algorithms run in $O(Gn^3)$ and $O(Hn^4)$ time for the first-order and second-order models respectively, where G and H are grammar constants.

3.2 Efficient Parsing

The efficiency of the parsing algorithm is important in applying the parsing model to test sentences, and also when training the model using discriminative methods. The grammar constants G and H introduced in the previous section are polynomial in factors such as the number of possible spines in the model, and the number of possible states in the finite-state automata implicit in the parsing algorithm. These constants are large, making exhaustive parsing very expensive.

To deal with this problem, we use a simple initial model to prune the search space of the more complex model. The first-stage model we use is a first-order dependency model, with labeled dependencies, as described in (McDonald et al., 2005). As described shortly, we will use this model to compute *marginal* scores for dependencies in both training and test sentences. A marginal score $\mu(\mathbf{x}, h, m, l)$ is a value between 0 and 1 that reflects the plausibility of a dependency for sentence \mathbf{x} with head-word x_h , modifier word x_m , and label l . In the first-stage pruning model the labels l are triples of non-terminals representing grammatical relations, as described in section 2.1 of this paper—for example, one possible label would be $\langle VP VBD NP \rangle$, and in general any triple of non-terminals is possible.

Given a sentence \mathbf{x} , and an index m of a word in that sentence, we define $D_{MAX}(\mathbf{x}, m)$ to be the

highest scoring dependency with m as a modifier:

$$\text{DMAX}(\mathbf{x}, m) = \max_{h,l} \mu(\mathbf{x}, h, m, l)$$

For a sentence \mathbf{x} , we then define the set of allowable dependencies to be

$$\pi(\mathbf{x}) = \{ \langle h, m, l \rangle : \mu(\mathbf{x}, h, m, l) \geq \alpha \text{DMAX}(\mathbf{x}, m) \}$$

where α is a constant dictating the beam size that is used (in our experiments we used $\alpha = 10^{-6}$).

The set $\pi(\mathbf{x})$ is used to restrict the set of possible parses under the full TAG-based model. In section 2.1 we described how the TAG model has dependency labels of the form $\langle \text{POS}, \text{A}, \eta_h, \eta_m, \text{L} \rangle$, and that there is a function GRM that maps labels of this form to triples of non-terminals. The basic idea of the pruned search is to only allow dependencies of the form $\langle h, m, \langle \text{POS}, \text{A}, \eta_h, \eta_m, \text{L} \rangle \rangle$ if the tuple $\langle h, m, \text{GRM}(\langle \text{POS}, \text{A}, \eta_h, \eta_m, \text{L} \rangle) \rangle$ is a member of $\pi(\mathbf{x})$, thus reducing the search space for the parser.

We now turn to how the marginals $\mu(\mathbf{x}, h, m, l)$ are defined and computed. A simple approach would be to use a conditional log-linear model (Lafferty et al., 2001), with features as defined by McDonald et al. (2005), to define a distribution $P(y|\mathbf{x})$ where the parse structures y are dependency structures with labels that are triples of non-terminals. In this case we could define

$$\mu(\mathbf{x}, h, m, l) = \sum_{y: \langle h, m, l \rangle \in y} P(y|\mathbf{x})$$

which can be computed with inside-outside style algorithms, applied to the data structures from (Eisner, 2000). The complexity of training and applying such a model is again $O(Gn^3)$, where G is the number of possible labels, and the number of possible labels (triples of non-terminals) is around $G = 1000$ in the case of treebank parsing; this value for G is still too large for the method to be efficient. Instead, we train three separate models μ_1 , μ_2 , and μ_3 for the three different positions in the non-terminal triples. We then take $\mu(\mathbf{x}, h, m, l)$ to be a product of these three models, for example we would calculate

$$\begin{aligned} \mu(\mathbf{x}, h, m, \langle \text{VP VBD NP} \rangle) &= \\ &\mu_1(\mathbf{x}, h, m, \langle \text{VP} \rangle) \times \mu_2(\mathbf{x}, h, m, \langle \text{VBD} \rangle) \\ &\times \mu_3(\mathbf{x}, h, m, \langle \text{NP} \rangle) \end{aligned}$$

Training the three models, and calculating the marginals, now has a grammar constant equal

to the number of non-terminals in the grammar, which is far more manageable. We use the algorithm described in (Globerson et al., 2007) to train the conditional log-linear model; this method was found to converge to a good model after 10 iterations over the training data.

4 Implementation Details

4.1 Features

Section 2.2 described the use of feature vectors associated with spines used in a derivation, together with first-order, sibling, and grandparent dependencies. The dependency features used in our experiments are closely related to the features described in (Carreras, 2007), which are an extension of the McDonald and Pereira (2006) features to cover grandparent dependencies in addition to first-order and sibling dependencies. The features take into account the identity of the labels l used in the derivations. The features could potentially look at any information in the labels, which are of the form $\langle \text{POS}, \text{A}, \eta_h, \eta_m, \text{L} \rangle$, but in our experiments, we map labels to a pair $(\text{GRM}(\langle \text{POS}, \text{A}, \eta_h, \eta_m, \text{L} \rangle), \text{A})$. Thus the label features are sensitive only to the triple of non-terminals corresponding to the grammatical relation involved in an adjunction, and a binary flag specifying whether the operation is s-adjunction or r-adjunction.

For the spine features $\mathbf{e}(\mathbf{x}, \langle i, \eta \rangle)$, we use feature templates that are sensitive to the identity of the spine η , together with contextual features of the string \mathbf{x} . These features consider the identity of the words and part-of-speech tags in a window that is centered on x_i and spans the range $x_{(i-2)} \cdots x_{(i+2)}$.

4.2 Extracting Derivations from Parse Trees

In the experiments in this paper, the following three-step process was used: (1) derivations were extracted from a training set drawn from the Penn WSJ treebank, and then used to train a parsing model; (2) the test data was parsed using the resulting model, giving a derivation for each test data sentence; (3) the resulting test-data derivations were mapped back to Penn-treebank style trees, using the method described in section 2.1. To achieve step (1), we first apply a set of head-finding rules which are similar to those described in (Collins, 1997). Once the head-finding rules have been applied, it is straightforward to extract

	precision	recall	F ₁
PPK07	–	–	88.3
FKM08	88.2	87.8	88.0
CH2000	89.5	89.6	89.6
CO2000	89.9	89.6	89.8
PK07	90.2	89.9	90.1
this paper	91.4	90.7	91.1
CJ05	–	–	91.4
H08	–	–	91.7
CO2000(s24)	89.6	88.6	89.1
this paper (s24)	91.1	89.9	90.5

Table 1: Results for different methods. PPK07, FKM08, CH2000, CO2000, PK07, CJ05 and H08 are results on section 23 of the Penn WSJ treebank, for the models of Petrov et al. (2007), Finkel et al. (2008), Charniak (2000), Collins (2000), Petrov and Klein (2007), Charniak and Johnson (2005), and Huang (2008). (CJ05 is the performance of an updated model at <http://www.cog.brown.edu/mj/software.htm>.) “s24” denotes results on section 24 of the treebank.

	s23	s24
KCC08 unlabeled	92.0	91.0
KCC08 labeled	92.5	91.7
this paper	93.5	92.5

Table 2: Table showing unlabeled dependency accuracy for sections 23 and 24 of the treebank, using the method of (Yamada and Matsumoto, 2003) to extract dependencies from parse trees from our model. KCC08 unlabeled is from (Koo et al., 2008), a model that has previously been shown to have higher accuracy than (McDonald and Pereira, 2006). KCC08 labeled is the labeled dependency parser from (Koo et al., 2008); here we only evaluate the unlabeled accuracy.

derivations from the Penn treebank trees.

Note that the mapping from parse trees to derivations is many-to-one: for example, the example trees in section 2.3 have structures that are as “flat” (have as few levels) as is possible, given the set D that is involved. Other similar trees, but with more VP levels, will give the same set D . However, this issue appears to be benign in the Penn WSJ treebank. For example, on section 22 of the treebank, if derivations are first extracted using the method described in this section, then mapped back to parse trees using the method described in section 2.3, the resulting parse trees score 100% precision and 99.81% recall in labeled constituent accuracy, indicating that very little information is lost in this process.

4.3 Part-of-Speech Tags, and Spines

Sentences in training, test, and development data are assumed to have part-of-speech (POS) tags. POS tags are used for two purposes: (1) in the features described above; and (2) to limit the set of allowable spines for each word during parsing. Specifically, for each POS tag we create a separate

α	1st stage		2nd stage		
	active	coverage	oracle F ₁	speed	F ₁
10^{-4}	0.07	97.7	97.0	5:15	91.1
10^{-5}	0.16	98.5	97.9	11:45	91.6
10^{-6}	0.34	99.0	98.5	21:50	92.0

Table 3: Effect of the beam size, controlled by α , on the performance of the parser on the development set (1,699 sentences). In each case α refers to the beam size used in both training and testing the model. “active”: percentage of dependencies that remain in the beam out of the total number of labeled dependencies (1,000 triple labels times 1,138,167 unlabeled dependencies); “coverage”: percentage of correct dependencies in the beam out of the total number of correct dependencies. “oracle F₁”: maximum achievable score of constituents, given the beam. “speed”: parsing time in *min:sec* for the TAG-based model (this figure does not include the time taken to calculate the marginals using the lower-order model); “F₁”: score of predicted constituents.

dictionary listing the spines that have been seen with this POS tag in training data; during parsing we only allow spines that are compatible with this dictionary. (For test or development data, we used the part-of-speech tags generated by the parser of (Collins, 1997). Future work should consider incorporating the tagging step within the model; it is not challenging to extend the model in this way.)

5 Experiments

Sections 2-21 of the Penn Wall Street Journal treebank were used as training data in our experiments, and section 22 was used as a development set. Sections 23 and 24 were used as test sets. The model was trained for 20 epochs with the averaged perceptron algorithm, with the development data performance being used to choose the best epoch. Table 1 shows the results for the method.

Our experiments show an improvement in performance over the results in (Collins, 2000; Charniak, 2000). We would argue that the Collins (2000) method is considerably more complex than ours, requiring a first-stage generative model, together with a reranking approach. The Charniak (2000) model is also arguably more complex, again using a carefully constructed generative model. The accuracy of our approach also shows some improvement over results in (Petrov and Klein, 2007). This work makes use of a PCFG with latent variables that is trained using a split/merge procedure together with the EM algorithm. This work is in many ways complementary to ours—for example, it does not make use of GLMs, dependency features, or of representations that go beyond PCFG productions—and

some combination of the two methods may give further gains.

Charniak and Johnson (2005), and Huang (2008), describe approaches that make use of non-local features in conjunction with the Charniak (2000) model; future work may consider extending our approach to include non-local features. Finally, other recent work (Petrov et al., 2007; Finkel et al., 2008) has had a similar goal of scaling GLMs to full syntactic parsing. These models make use of PCFG representations, but do not explicitly model bigram or trigram dependencies. The results in this work (88.3%/88.0% F_1) are lower than our F_1 score of 91.1%; this is evidence of the benefits of the richer representations enabled by our approach.

Table 2 shows the accuracy of the model in recovering unlabeled dependencies. The method shows improvements over the method described in (Koo et al., 2008), which is a state-of-the-art second-order dependency parser similar to that of (McDonald and Pereira, 2006), suggesting that the incorporation of constituent structure can improve dependency accuracy.

Table 3 shows the effect of the beam-size on the accuracy and speed of the parser on the development set. With the beam setting used in our experiments ($\alpha = 10^{-6}$), only 0.34% of possible dependencies are considered by the TAG-based model, but 99% of all correct dependencies are included. At this beam size the best possible F_1 constituent score is 98.5. Tighter beams lead to faster parsing times, with slight drops in accuracy.

6 Conclusions

We have described an efficient and accurate parser for constituent parsing. A key to the approach has been to use a splittable grammar that allows efficient dynamic programming algorithms, in combination with pruning using a lower-order model. The method allows relatively easy incorporation of features; future work should leverage this in producing more accurate parsers, and in applying the parser to different languages or domains.

Acknowledgments X. Carreras was supported by the Catalan Ministry of Innovation, Universities and Enterprise, by the GALE program of DARPA, Contract No. HR0011-06-C-0022, and by a grant from NTT, Agmt. Dtd. 6/21/1998. T. Koo was funded by NSF grant IIS-0415030. M. Collins was funded by NSF grant IIS-0347631 and DARPA contract No. HR0011-06-C-0022. Thanks to Jenny Rose Finkel for suggesting that we evaluate dependency parsing accuracies.

References

- Altun, Y., I. Tsochantaridis, and T. Hofmann. 2003. Hidden markov support vector machines. In *ICML*.
- Carreras, X. 2007. Experiments with a higher-order projective dependency parser. In *Proc. EMNLP-CoNLL Shared Task*.
- Charniak, E. and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. ACL*.
- Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proc. NAACL*.
- Chiang, D. 2003. Statistical parsing with an automatically extracted tree adjoining grammar. In Bod, R., R. Scha, and K. Sima'an, editors, *Data Oriented Parsing*, pages 299–316. CSLI Publications.
- Clark, S. and J. R. Curran. 2004. Parsing the wsj using ccg and log-linear models. In *Proc. ACL*.
- Clark, Stephen and James R. Curran. 2007. Perceptron training for a wide-coverage lexicalized-grammar parser. In *Proc. ACL Workshop on Deep Linguistic Processing*.
- Collins, M. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. ACL*.
- Collins, M. 2000. Discriminative reranking for natural language parsing. In *Proc. ICML*.
- Collins, M. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*.
- Eisner, J. 2000. Bilexical grammars and their cubic-time parsing algorithms. In Bunt, H. C. and A. Nijholt, editors, *New Developments in Natural Language Parsing*, pages 29–62. Kluwer Academic Publishers.
- Finkel, J. R., A. Kleeman, and C. D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proc. ACL/HLT*.
- Globerson, A., T. Koo, X. Carreras, and M. Collins. 2007. Exponentiated gradient algorithms for log-linear structured prediction. In *Proc. ICML*.
- Huang, L. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. ACL/HLT*.
- Johnson, M., S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic unification-based grammars. In *Proc. ACL*.
- Koo, Terry, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. ACL/HLT*.
- Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.
- McDonald, R. and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. EACL*.
- McDonald, R., K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. ACL*.
- Petrov, S. and D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. of HLT-NAACL*.
- Petrov, S., A. Pauls, and D. Klein. 2007. Discriminative log-linear grammars with latent variables. In *Proc. NIPS*.
- Ratnaparkhi, A., S. Roukos, and R. Ward. 1994. A maximum entropy model for parsing. In *Proc. ICSLP*.
- Sagae, Kenji, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Hpsg parsing with shallow dependency constraints. In *Proc. ACL*, pages 624–631.
- Shen, L. and A.K. Joshi. 2005. Incremental ltag parsing. In *Proc HLT-EMNLP*.
- Taskar, B., D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *Proceedings of the EMNLP-2004*.
- Yamada, H. and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. IWPT*.

A Fast Boosting-based Learner for Feature-Rich Tagging and Chunking

Tomoya Iwakura Seishi Okamoto

Fujitsu Laboratories Ltd.

1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki 211-8588, Japan

{iwakura.tomoya, seishi}@jp.fujitsu.com

Abstract

Combination of features contributes to a significant improvement in accuracy on tasks such as part-of-speech (POS) tagging and text chunking, compared with using atomic features. However, selecting combination of features on learning with large-scale and feature-rich training data requires long training time. We propose a fast boosting-based algorithm for learning rules represented by combination of features. Our algorithm constructs a set of rules by repeating the process to select several rules from a small proportion of candidate rules. The candidate rules are generated from a subset of all the features with a technique similar to beam search. Then we propose POS tagging and text chunking based on our learning algorithm. Our tagger and chunker use candidate POS tags or chunk tags of each word collected from automatically tagged data. We evaluate our methods with English POS tagging and text chunking. The experimental results show that the training time of our algorithm are about 50 times faster than Support Vector Machines with polynomial kernel on the average while maintaining state-of-the-art accuracy and faster classification speed.

1 Introduction

Several boosting-based learning algorithms have been applied to Natural Language Processing problems successfully. These include text categorization (Schapire and Singer, 2000), Natural Language Parsing (Collins and Koo, 2005), English syntactic chunking (Kudo et al., 2005) and so on.

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

Furthermore, classifiers based on boosting-based learners have shown fast classification speed (Kudo et al., 2005).

However, boosting-based learning algorithms require long training time. One of the reasons is that boosting is a method to create a final hypothesis by repeatedly generating a weak hypothesis in each training iteration with a given weak learner. These weak hypotheses are combined as the final hypothesis. Furthermore, the training speed of boosting-based algorithms becomes more of a problem when considering combination of features that contributes to improvement in accuracy.

This paper proposes a fast boosting-based algorithm for learning rules represented by combination of features. Our learning algorithm uses the following methods to learn rules from large-scale training samples in a short time while maintaining accuracy; 1) Using a rule learner that learns several rules as our weak learner while ensuring a reduction in the theoretical upper bound of the training error of a boosting algorithm, 2) Repeating to learn rules from a small proportion of candidate rules that are generated from a subset of all the features with a technique similar to beam search, 3) Changing subsets of features used by weak learner dynamically for alleviating overfitting.

We also propose feature-rich POS tagging and text chunking based on our learning algorithm. Our POS tagger and text chunker use candidate tags of each word obtained from automatically tagged data as features.

The experimental results with English POS tagging and text chunking show drastically improvement of training speeds while maintaining competitive accuracy compared with previous best results and fast classification speeds.

2 Boosting-based Learner

2.1 Preliminaries

We describe the problem treated by our boosting-based learner as follows. Let \mathcal{X} be the set of examples and \mathcal{Y} be a set of labels $\{-1, +1\}$. Let $\mathcal{F} = \{f_1, f_2, \dots, f_M\}$ be M types of features represented by strings. Let S be a set of training sam-

```

##  $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m : \mathbf{x}_i \subseteq \mathcal{X}, y_i \in \{\pm 1\}$ 
## a smoothing value  $\varepsilon = 1$ 
## rule number  $r$ : the initial value is 1.
Initialize: For  $i=1, \dots, m$ :  $w_{1,i} = \exp(\frac{1}{2} \log(\frac{W_{+1}}{W_{-1}}))$ ;
While ( $r < R$ )
## Train weak-learner using ( $S, \{w_{r,i}\}_{i=1}^m$ )
## Get  $\nu$  types of rules:  $\{\mathbf{f}_j\}_{j=1}^\nu$ 
 $\{\mathbf{f}_j\}_{j=1}^\nu \leftarrow \text{weak-learner}(S, \{w_{r,i}\}_{i=1}^m)$ ;
## Update weights with confidence value
ForEach  $\mathbf{f} \in \{\mathbf{f}_j\}_{j=1}^\nu$ 
 $c = \frac{1}{2} \log(\frac{W_{r,+1}(\mathbf{f}) + \varepsilon}{W_{r,-1}(\mathbf{f}) + \varepsilon})$ 
For  $i=1, \dots, m$ :  $w_{r+1,i} = w_{r,i} \exp(-y_i h_{(\mathbf{f}, c)})$ 
 $\mathbf{f}_r = \mathbf{f}; c_r = c; r++$ ;
endForEach
endWhile
Output:  $F(\mathbf{x}) = \text{sign}(\log(\frac{W_{+1}}{W_{-1}}) + \sum_{r=1}^R h_{(\mathbf{f}_r, c_r)}(\mathbf{x}))$ 

```

Figure 1: A generalized version of our learner

ples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, where each example $\mathbf{x}_i \in \mathcal{X}$ consists of features in \mathcal{F} , which we call a feature-set, and $y_i \in \mathcal{Y}$ is a class label. The goal is to induce a mapping

$$F : \mathcal{X} \rightarrow \mathcal{Y}$$

from S .

Let $|\mathbf{x}_i|$ ($0 < |\mathbf{x}_i| \leq M$) be the number of features included in a feature-set \mathbf{x}_i , which we call the size of \mathbf{x}_i , and $x_{i,j} \in \mathcal{F}$ ($1 \leq j \leq |\mathbf{x}_i|$) be a feature included in \mathbf{x}_i .¹ We call a feature-set of size k a k -feature-set. Then we define subsets of feature-sets as follows.

Definition 1 *Subsets of feature-sets*

If a feature-set \mathbf{x}_j contains all the features in a feature-set \mathbf{x}_i , then we call \mathbf{x}_i is a subset of \mathbf{x}_j and denote it as

$$\mathbf{x}_i \subseteq \mathbf{x}_j.$$

Then we define weak hypothesis based on the idea of the real-valued predictions and abstaining (RVPA, for short) (Schapire and Singer, 2000).²

Definition 2 *Weak hypothesis for feature-sets*

Let \mathbf{f} be a feature-set, called a rule, \mathbf{x} be a feature-set, and c be a real number, called a confidence value, then a weak-hypothesis for feature-sets is defined as

$$h_{(\mathbf{f}, c)}(\mathbf{x}) = \begin{cases} c & \mathbf{f} \subseteq \mathbf{x} \\ 0 & \text{otherwise} \end{cases}$$

¹Our learner can handle binary vectors as in (Morishita, 2002). When our learner treats binary vectors for M attributes $\{\mathbf{X}_1, \dots, \mathbf{X}_m\}$, the learner converts each vector to the corresponding feature-set as $\mathbf{x}_i \leftarrow \{f_i | X_{i,j} \in \mathbf{X}_i \wedge X_{i,j} = 1\}$ ($1 \leq i \leq m, 1 \leq j \leq M$).

²We use the RVPA because training with RVPA is faster than training with Real-valued-predictions (RVP) while maintaining competitive accuracy (Schapire and Singer, 2000). The idea of RVP is to output a confidence value for samples which do not satisfy the given condition too.

2.2 Boosting-based Rule Learning

Our boosting-based learner selects R types of rules for creating a final hypothesis F on several training iterations. The F is defined as

$$F(\mathbf{x}) = \text{sign}(\sum_{r=1}^R h_{(\mathbf{f}_r, c_r)}(\mathbf{x})).$$

We use a learning algorithm that generates several rules from a given training samples $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ and weights over samples $\{w_{r,1}, \dots, w_{r,m}\}$ as input of our weak learner. $w_{r,i}$ is the weight of sample number i after selecting $r - 1$ types of rules, where $0 < w_{r,i}, 1 \leq i \leq m$ and $1 \leq r \leq R$.

Given such input, the weak learner selects ν types of rules $\{\mathbf{f}_j\}_{j=1}^\nu$ ($\mathbf{f}_j \subseteq \mathcal{F}$) with *gain*:

$$\text{gain}(\mathbf{f}) \stackrel{\text{def}}{=} |\sqrt{W_{r,+1}(\mathbf{f})} - \sqrt{W_{r,-1}(\mathbf{f})}|,$$

where \mathbf{f} is a feature-set, and $W_{r,y}(\mathbf{f})$ is

$$W_{r,y}(\mathbf{f}) = \sum_{i=1}^m w_{r,i} [[\mathbf{f} \subseteq \mathbf{x}_i \wedge y_i = y]],$$

and $[[\pi]]$ is 1 if a proposition π holds and 0 otherwise.

The weak learner selects a feature-set having the highest *gain* as the first rule, and the weak learner finally selects ν types of feature-sets having *gain* in top ν as $\{\mathbf{f}_j\}_{j=1}^\nu$ at each boosting iteration.

Then the boosting-based learner calculates the confidence value of each \mathbf{f} in $\{\mathbf{f}_j\}_{j=1}^\nu$ and updates the weight of each sample. The confidence value c_j for \mathbf{f}_j is defined as

$$c_j = \frac{1}{2} \log(\frac{W_{r,+1}(\mathbf{f}_j)}{W_{r,-1}(\mathbf{f}_j)}).$$

After the calculation of c_j for \mathbf{f}_j , the learner updates the weight of each sample with

$$w_{r+1,i} = w_{r,i} \exp(-y_i h_{(\mathbf{f}_j, c_j)}). \quad (1)$$

Then the learner adds (\mathbf{f}_j, c_j) to F as the r -th rule and its confidence value.³ When we calculate the confidence value c_{j+1} for \mathbf{f}_{j+1} , we use $\{w_{r+1,1}, \dots, w_{r+1,m}\}$. The learner adds $(\mathbf{f}_{j+1}, c_{j+1})$ to F as the $r+1$ -th rule and confidence value.

After the updates of weights with $\{\mathbf{f}_j\}_{j=1}^\nu$, the learner starts the next boosting iteration. The learner continues training until obtaining R rules.

Our boosting-based algorithm differs from the other boosting algorithms in the number of rules learned at each iteration. The other boosting-based algorithms usually learn a rule at each iteration

³Eq. (1) is the update of the AdaBoost used in ADTrees learning algorithm (Freund and Mason, 1999). We use this AdaBoost by the following two reasons. 1) The paper (Iwakura and Okamoto, 2007) showed that the accuracy of text chunking with the AdaBoost of ADTrees is slightly higher than text chunking with the AdaBoost of BoosTexter for RVPA (Schapire and Singer, 2000), 2) We expect the AdaBoost of ADTrees can realize faster training because this AdaBoost does not normalize weights at each update compared with the AdaBoost of BoosTexter normalizes weights at each iteration.

```

## sortByW( $\mathcal{F}, fq$ ): Sort features ( $f \in \mathcal{F}$ )
## in ascending order based on weights of features
## ( $a \% b$ ): Return the remainder of ( $a \div b$ )
##  $|B|$ -buckets:  $B = \{B[0], \dots, B[|B| - 1]\}$ 
procedure distFT( $S, |B|$ )
## Calculate the weight of each feature
ForEach ( $f \in \mathcal{F}$ )  $W_r(f) = \sum_{i=1}^m w_{r,i}[\{f\} \subseteq \mathbf{x}_i]$ 
## Sort features based on their weights and
## store the results in  $F_s$ 
 $F_s \leftarrow \text{sortByW}(\mathcal{F}, W_r)$ 
## Distribute features to buckets
For  $i=0 \dots M$ :  $B[(i \% |B|)] = (B[(i \% |B|)] \cup F_s[i])$ 
return  $B$ 

```

Figure 2: Distribute features to buckets based on weights

(Schapire and Singer, 2000; Freund and Mason, 1999). Despite the difference, our boosting-based algorithm ensures a reduction in the theoretical upper bound of training error of the AdaBoost. We list the detailed explanation in Appendix.A.

Figure 1 shows an overview of our boosting-based rule learner. To avoid to happen that $W_{r,+1}(\mathbf{f})$ or $W_{r,-1}(\mathbf{f})$ is very small or even zero, we use the smoothed values ε (Schapire and Singer, 1999). Furthermore, to reflect imbalance class distribution, we use the default rule (Freund and Mason, 1999), defined as $\frac{1}{2} \log(\frac{W_{+1}}{W_{-1}})$, where $W_y = \sum_{i=1}^m [[y_i = y]]$ for $y \in \{\pm 1\}$. The initial weights are defined with the default rule.

3 Fast Rule Learner

3.1 Generating Candidate Rules

We use a method to generate candidate rules without duplication (Iwakura and Okamoto, 2007). We denote $\mathbf{f}' = \mathbf{f} + f$ as the generation of $k + 1$ -feature-set \mathbf{f}' consisting of a feature f and a k -feature-set \mathbf{f} . Let $ID(f)$ be the integer corresponding to f , called *id*, and ϕ be 0-feature-set. Then we define *gen* generating a feature-set as

$$\text{gen}(\mathbf{f}, f) = \begin{cases} \mathbf{f} + f & \text{if } ID(f) > \max_{f' \in \mathcal{F}} ID(f') \\ \phi & \text{otherwise} \end{cases}.$$

We assign smaller integer to more infrequent features as *id*. If there are features having the same frequency, we assign *id* to each feature with lexicographic order of features. Training based on this candidate generation showed faster training speed than generating candidates by an arbitrary order (Iwakura and Okamoto, 2007).

3.2 Training with Redistributed Features

We propose a method for learning rules by repeating to select a rule from a small portion of candidate rules. We evaluated the effectiveness of four types of methods to learn a rule from a subset of features on boosting-based learners with a text chunking task (Iwakura and Okamoto, 2007). The results showed that Frequency-based distribution (*F-dist*) has shown the best accuracy. *F-dist*

```

##  $F_k$ : A set of  $k$ -feature-sets
##  $\mathcal{R}_o$ :  $\nu$  optimal rules (feature-sets)
##  $R_{k,\omega}$ :  $\omega$   $k$ -feature-sets for generating candidates
## selectNBest( $\mathcal{R}, n, S, W_r$ ):  $n$  best rules from  $\mathcal{R}$ 
## with gain on  $\{w_{i,r}\}_{i=1}^m$  and training samples  $S$ 
procedure weak-learner( $F_k, S, W_r$ )
##  $\nu$  best feature-sets as rules
 $\mathcal{R}_o = \text{selectNBest}(\mathcal{R}_o \cup F_k, \nu, S, W_r)$ 
if ( $\zeta \leq k$ ) return  $\mathcal{R}_o$ ; ## Size constraint
##  $\omega$  best feature-sets in  $F_k$  for generating candidates
 $R_{k,\omega} = \text{selectNBest}(F_k, \omega, S, W_r)$ 
 $\tau = \min_{\mathbf{f} \in \mathcal{R}_o} \text{gain}(\mathbf{f})$ ; ## The gain of  $\nu$ -th optimal rule
ForEach ( $\mathbf{f}_k \in R_{k,\omega}$ )
if ( $u(\mathbf{f}_k) < \tau$ ) continue; ## Upper bound of gain
ForEach ( $f \in \mathcal{F}$ ) ## Generate candidates
 $\mathbf{f}_{k+1} = \text{gen}(\mathbf{f}_k, f)$ ;
if ( $\xi \leq \sum_{i=1}^m [[\mathbf{f}_{k+1} \subseteq \mathbf{x}_i]]$ )  $F_{k+1} = (F_{k+1} \cup \mathbf{f}_{k+1})$ ;
end ForEach
end ForEach
return weak-learner( $F_{k+1}, S, W_r$ );

```

Figure 3: Find optimal feature-sets with given weights

distributes features to subsets of features, called buckets, based on frequencies of features.

However, we guess training using a subset of features depends on how to distribute features to buckets like online learning algorithms that generally depend on the order of the training examples (Kazama and Torisawa, 2007).

To alleviate the dependency on selected buckets, we propose a method that redistributes features, called Weight-based distribution (*W-dist*). *W-dist* redistributes features to buckets based on the weight of feature defined as

$$W_r(f) = \sum_{i=1}^m w_{r,i}[\{f\} \subseteq \mathbf{x}_i]$$

for each $f \in \mathcal{F}$ after examining all buckets. Figure 2 describes an overview of *W-dist*.

3.3 Weak Learner for Learning Several Rules

We propose a weak learner that learns several rules from a small portion of candidate rules.

Figure 3 describes an overview of the weak learner. At each iteration, one of the $|B|$ -buckets is given as an initial 1-feature-sets F_1 . The weak learner finds ν best feature-sets as rules from candidates consisting of F_1 and feature-sets generated from F_1 . The weak learner generates candidates k -feature-sets ($1 < k$) from ω best $(k-1)$ -feature-sets in F_{k-1} with *gain*.

We also use the following pruning techniques (Morishita, 2002; Kudo et al., 2005).

- **Frequency constraint:** We examine candidates seen on at least ξ different examples.
- **Size constraint:** We examine candidates whose size is no greater than a size threshold ζ .
- **Upper bound of gain:** We use the upper bound of gain defined as

$$u(\mathbf{f}) \stackrel{\text{def}}{=} \max(\sqrt{W_{r,+1}(\mathbf{f})}, \sqrt{W_{r,-1}(\mathbf{f})}).$$

For any feature-set $\mathbf{f}' \subseteq \mathcal{F}$, which contains \mathbf{f} (i.e. $\mathbf{f} \subseteq \mathbf{f}'$), the *gain*(\mathbf{f}') is bounded under $u(\mathbf{f})$, since $0 \leq W_{r,y}(\mathbf{f}') \leq W_{r,y}(\mathbf{f})$ for $y \in \{\pm 1\}$. Thus, if $u(\mathbf{f})$

```

##  $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m : \mathbf{x}_i \subseteq \mathcal{X}, y_i \in \{+1\}$ 
##  $W_r = \{w_{r,i}\}_{i=1}^m$ : Weights of samples after learning
##  $r$  types of rules.  $w_{1,i} = 1$  ( $1 \leq i \leq m$ )
##  $|B|$ : The size of bucket  $B = \{B[0], \dots, B[|B| - 1]\}$ 
##  $b, r$ : The current bucket and rule number
procedure AdaBoost.SDF()
 $B = \text{distFT}(S, |B|)$ ; ## Distributing features into  $B$ 
## Initialize values and weights:
 $r = 1; b = 0; c_0 = \frac{1}{2} \log(\frac{W_{+1}}{W_{-1}})$ ;
For  $i = 1, \dots, m$ :  $w_{1,i} = \exp(c_0)$ ;
While ( $r \leq R$ ) ## Learning  $R$  types of rules
## Select  $\nu$  rules and increment bucket id  $b$ 
 $\mathcal{R} = \text{weak-learner}(B[b], S, W_r); b++$ ;
Foreach ( $f \in \mathcal{R}$ ) ## Update weights with each rule
 $c = \frac{1}{2} \log(\frac{W_{r,+1}(f)+1}{W_{r,-1}(f)+1})$ ;
For  $i=1, \dots, m$   $w_{r+1,i} = w_{r,i} \exp(-y_i h_{(f,c)})$ ;
 $f_r = f; c_r = c; r++$ ;
end Foreach
if ( $b == |B|$ ) ## Redistribution
 $B = \text{distFT}(S, |B|); b=0$ ;
end if
end While
return  $F(\mathbf{x}) = \text{sign}(c_0 + \sum_{r=1}^R h_{(f_r, c_r)}(\mathbf{x}))$ 

```

Figure 4: An overview of AdaBoost.SDF

- words, words that are turned into all capitalized, prefixes and suffixes (up to 4) in a 7-word window.
- labels assigned to three words on the right.
- whether the current word has a hyphen, a number, a capital letter
- whether the current word is all capital, all small
- candidate POS tags of words in a 7-word window

Figure 5: Feature types for POS tagging

is less than the *gain* of the current optimal rule τ , candidates containing \mathbf{f} are safely pruned.

Figure 4 describes an overview of our algorithm, which we call AdaBoost for a weak learner learning Several rules from *Distributed Features (AdaBoost.SDF, for short)*.

The training of AdaBoost.SDF with ($\nu = 1, \omega = \infty, 1 < |B|$) is equivalent to the approach of AdaBoost.DF (Iwakura and Okamoto, 2007). If we use ($|B| = 1, \nu = 1$), AdaBoost.SDF examines all features on every iteration like (Freund and Mason, 1999; Schapire and Singer, 2000).

4 POS tagging and Text Chunking

4.1 English POS Tagging

We used the Penn Wall Street Journal treebank (Marcus et al., 1994). We split the treebank into training (sections 0-18), development (sections 19-21) and test (sections 22-24) as in (Collins, 2002). We used the following candidate POS tags, called candidate feature, in addition to commonly used features (Giménez and Màrquez, 2003; Toutanova et al., 2003) shown in Figure 5.

We collect candidate POS tags of each word from the automatically tagged corpus provided for the shared task of English Named Entity recognition in CoNLL 2003.⁴ The corpus includes 17,003,926 words with POS tags and chunk tags

⁴<http://www.cnts.ua.ac.be/conll2003/ner/>

- words and POS tags in a 5-word window.
- labels assigned to two words on the right.
- candidate chunk tags of words in a 5-word window

Figure 6: Feature types for text chunking

annotated by a POS tagger and a text chunker. Thus, the corpus includes wrong POS tags and chunk tags.

We collected candidate POS tags of words that appear more than 9 times in the corpus. We express these candidates with one of the following ranges decided by their frequency f_q ; $10 \leq f_q < 100$, $100 \leq f_q < 1000$ and $1000 \leq f_q$.

For example, we express 'work' annotated as NN 2000 times like "1000≤NN". If 'work' is current word, we add 1000≤NN as a candidate POS tag feature of the current word. If 'work' appears the next of the current word, we add 1000≤NN as a candidate POS tag of the next word.

4.2 Text Chunking

We used the data prepared for CoNLL-2000 shared tasks.⁵ This task aims to identify 10 types of chunks, such as, NP, VP and PP, and so on.

The data consists of subsets of Penn Wall Street Journal treebank; training (sections 15-18) and test (section 20). We prepared the development set from section 21 of the treebank as in (Tsuruoka and Tsujii, 2005).⁶

Each base phrase consists of one word or more. To identify word chunks, we use IOE2 representation. The chunks are represented by the following tags: E-X is used for end word of a chunk of class X. I-X is used for non-end word in an X chunk. O is used for word outside of any chunk.

For instance, "[He] (NP) [reckons] (VP) [the current account deficit] (NP)..." is represented by IOE2 as follows; "He/E-NP reckons/E-VP the/I-NP current/I-NP account/I-NP deficit/E-NP".

We used features shown in Figure 6. We collected the followings as candidate chunk tags from the same automatically tagged corpus used in POS tagging.

- Candidate tags expressed with frequency information as in POS tagging
- The ranking of each candidate decided by frequencies in the automatically tagged data
- Candidate tags of each word

For example, if we collect "work" annotated as I-NP 2000 times and as E-VP 100 time, we generate the following candidate features for "work"; $1000 \leq \text{I-NP}$, $100 \leq \text{E-VP} < 1000$, rank:I-NP=1 rank:E-NP=2, candidate=I-NP and candidate=E-VP.

⁵<http://lcg-www.uia.ac.be/conll2000/chunking/>

⁶We used http://ilk.uvt.nl/~sabine/chunklink/chunklink.2-2-2000_for.conll.pl for creating development data.

Table 1: Training data for experiments: # of S, M , # of cl and av. # of ft indicate the number samples, the distinct number of feature types, the number of class in each data set, and the average number of features, respectively. POS and ETC indicate POS-tagging and text chunking. The “-c” indicates using candidate features collected from parsed unlabeled data.

data	# of S	M	# of cl	av. # of ft
POS	912,344	579,052	45	22.09
POS-c	912,344	579,793	45	35.39
ETC	211,727	92,825	22	11.37
ETC-c	211,727	93,333	22	45.49

We converted the chunk representation of the automatically tagged corpus to IOE2 and we collected chunk tags of each word appearing more than nine times.

4.3 Applying AdaBoost.SDF

AdaBoost.SDF treats the binary classification problem. To extend AdaBoost.SDF to multi-class, we used the one-vs-the-rest method.

To identify proper tag sequences, we use Viterbi search. We map the confidence value of each classifier into the range of 0 to 1 with sigmoid function⁷, and select a tag sequence which maximizes the sum of those log values by Viterbi search.

5 Experiments

5.1 Experimental Settings

We compared AdaBoost.SDF with Support Vector Machines (SVM). SVM has shown good performance on POS tagging (Giménez and Màrquez, 2003) and Text Chunking (Kudo and Matsumoto, 2001). Furthermore, SVM with polynomial kernel implicitly expands all feature combinations without increasing the computational costs. Thus, we compared AdaBoost.SDF with SVM.⁸

To evaluate the effectiveness of candidate features, we examined two types of experiments with candidate features and without them. We list the statics of training sets in Table 1.

We tested $R=100,000$, $|B|=1,000$, $\nu = \{1,10,100\}$, $\omega=\{1,10,100,\infty\}$, $\zeta=\{1,2,3\}$, and $\xi=\{1,5\}$ for AdaBoost.SDF. We tested the soft margin parameter $C=\{0.1,1,10\}$ and the kernel degree $d=\{1,2,3\}$ for SVM.⁹

We used the followings for comparison; *Training time* is time to learn 100,000 rules. *Best training time* is time for generating rules to show the best F-measure ($F_{\beta=1}$) on development data. *Accuracy* is $F_{\beta=1}$ on a test data with the rules at *best training time*.

⁷ $s(X) = 1/(1 + \exp(-\beta X))$, where $X = F(\mathbf{x})$ is a output of a classifier. We used $\beta=5$ in this experiment.

⁸We used TinySVM (<http://chases.org/~taku/software/TinySVM/>).

⁹We used machines with 2.66 GHz QuadCore Intel Xeon and 10 GB of memory for all the experiments.

Table 2: Experimental results of POS tagging and Text Chunking (TC) with candidate features. F and time indicate the average $F_{\beta=1}$ of test data and time (hour) to learn 100,000 rules for all classes with F -*dist*. These results are listed separately with respect to each $\xi = \{1, 5\}$.

ν	POS($\xi = 1$)		POS($\xi = 5$)		TC($\xi = 1$)		TC($\xi = 5$)	
	F	time	F	time	F	time	F	time
1	97.27	196.3	97.23	195.7	93.98	145.3	93.95	155.8
10	97.23	23.05	97.17	22.35	93.96	2.69	93.88	2.70
100	96.82	2.99	96.83	2.91	93.16	0.74	93.14	0.56

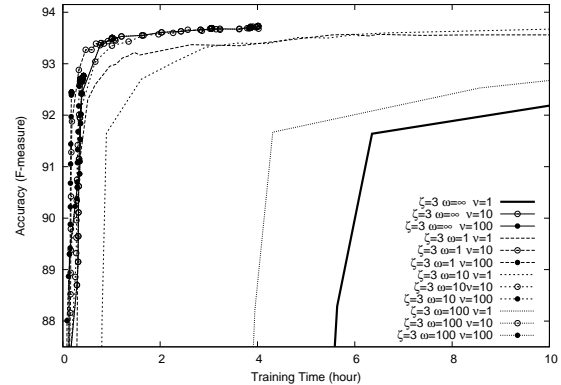


Figure 7: Accuracy on development data of Text Chunking ($\zeta = 3$) obtained with parsers based on F -*dist*. We measured accuracy obtained with rules at each training time. The widest line is AdaBoost.SDF ($\nu=1, \omega=\infty$). The others are AdaBoost.SDF ($\nu=10$ (\circ), $\nu=100$ (\bullet), $\nu=1$ & $\omega=\{1,10,100\}$).

5.2 Effectiveness of Several Rule Learning

Table 2 shows average accuracy and training time. We used F -*dist* as the distribution method. These average accuracy obtained with rules learned by AdaBoost.SDF ($\nu=10$) on both tasks are competitive with the average accuracy obtained with rules learned by AdaBoost.SDF ($\nu=1$). These results have shown that learning several rules at each iteration contributes significant improvement of training time. These results have also shown that the learning several rule at each iteration methods are more efficient than training by just using the frequency constraint ξ .

Figure 7 shows a snapshot for accuracy obtained with chunkers using different number of rules. This graph shows that chunkers based on AdaBoost.SDF ($\nu=10,100$) and AdaBoost.SDF ($\nu=1, \omega=\{1,10,100\}$) have shown better accuracy than chunkers based on AdaBoost.SDF ($\nu=1, \omega=\infty$) at each training time. These result have shown that learning several rules at each iteration and learning combination of features as rules with a technique similar to beam search are effective in improving training time while giving a better convergence.

Figure 7 also implies that taggers and chunkers based on AdaBoost.SDF ($\nu=100$) will show better or competitive accuracy than accuracy of the others by increasing numbers of rules to be learned while maintaining faster convergence speed.

Table 3: Experimental results on POS tagging and Text Chunking. *Accuracies* ($F_{\beta=1}$) on test data and training time (hour) of AdaBoost.SDF are averages of $\omega=\{1,10,100,\infty\}$ for each ζ with F -*dist* and $\xi = 1$. $F_{\beta=1}$ and time (hour) of SVMs are averages of $C=\{0.1,1,10\}$ for each kernel parameter d .

POS tagging without candidate features						
Alg. / ζ (d)	1		2		3	
	$F_{\beta=1}$	time	$F_{\beta=1}$	time	$F_{\beta=1}$	time
$\nu=1$	96.96	5.09	97.10	27.90	97.10	30.92
$\nu=10$	96.89	0.79	97.12	4.56	97.07	4.74
$\nu=100$	96.57	0.10	96.82	0.81	96.73	0.81
SVMs	96.60	101.63	97.15	166.76	96.93	625.32

POS tagging with candidate features						
Alg. / ζ (d)	1		2		3	
	$F_{\beta=1}$	time	$F_{\beta=1}$	time	$F_{\beta=1}$	time
$\nu=1$	97.06	6.65	97.30	109.20	97.29	330.82
$\nu=10$	96.98	1.27	97.29	13.26	97.23	38.27
$\nu=100$	96.61	0.14	96.93	1.64	96.76	5.05
SVMs	96.76	170.24	97.31	206.39	97.23	1346.04

Text Chunking without candidate features						
Alg. / ζ (d)	1		2		3	
	$F_{\beta=1}$	time	$F_{\beta=1}$	time	$F_{\beta=1}$	time
$\nu=1$	92.50	0.12	93.60	0.26	93.47	0.41
$\nu=10$	92.34	0.02	93.50	0.05	93.39	0.07
$\nu=100$	89.70	0.008	92.31	0.02	92.03	0.02
SVMs	92.14	8.55	93.91	7.38	93.49	9.82

Text Chunking with candidate features						
Alg. / ζ (d)	1		2		3	
	$F_{\beta=1}$	time	$F_{\beta=1}$	time	$F_{\beta=1}$	time
$\nu=1$	92.89	0.25	94.19	26.10	94.04	300.77
$\nu=10$	92.85	0.04	94.11	2.97	94.08	3.06
$\nu=100$	91.99	0.01	93.37	0.32	93.24	0.34
SVMs	92.77	12.74	94.31	9.63	94.20	49.27

5.3 Comparison with SVM

Table 3 lists average accuracy and training time on POS tagging and text chunking with respect to each (ν, ζ) for AdaBoost.SDF and d for SVM. AdaBoost.SDF with $\nu=10$ and $\nu=100$ have shown much faster training speeds than SVM and AdaBoost.SDF ($\nu=1, \omega=\infty$) that is equivalent to the AdaBoost.DF (Iwakura and Okamoto, 2007).

Furthermore, the accuracy of taggers and chunkers based on AdaBoost.SDF ($\nu=10$) have shown competitive accuracy with those of SVM-based and AdaBoost.DF-based taggers and chunkers. AdaBoost.SDF ($\nu=10$) showed about 6 and 54 times faster training speeds than those of AdaBoost.DF on the average in POS tagging and text chunking. AdaBoost.SDF ($\nu=10$) showed about 147 and 9 times faster training speeds than the training speeds of SVM on the average of POS tagging and text chunking. On the average of the both tasks, AdaBoost.SDF ($\nu=10$) showed about 25 and 50 times faster training speed than AdaBoost.DF and SVM. These results have shown that AdaBoost.SDF with a moderate parameter ν can improve training time drastically while maintaining accuracy.

These results in Table 3 have also shown that rules represented by combination of features and the candidate features collected from automatically tagged data contribute to improved accuracy.

5.4 Effectiveness of Redistribution

We compared $F_{\beta=1}$ and *best training time* of F -*dist* and W -*dist*. We used $\zeta = 2$ that has shown

Table 4: Results obtained with taggers and chunkers based on F -*dist* and W -*dist*. These results obtained with taggers and chunkers trained with $\omega = \{1, 10, 100, \infty\}$ and $\zeta = 2$. F and time indicate average $F_{\beta=1}$ on test data and average *best training time*.

POS tagging with F - <i>dist</i>								
ν	$\omega=1$		$\omega=10$		$\omega=100$		$\omega=\infty$	
	F	time	F	time	F	time	F	time
1	97.31	30.03	97.31	64.25	97.32	142.9	97.26	89.59
10	97.26	3.21	97.32	9.57	97.30	15.54	97.30	19.64
100	96.86	0.62	96.95	1.32	96.95	2.13	96.96	2.43

POS tagging with W - <i>dist</i>								
ν	$\omega=1$		$\omega=10$		$\omega=100$		$\omega=\infty$	
	F	time	F	time	F	time	F	time
1	97.32	29.96	97.31	57.05	97.31	163.2	97.32	98.71
10	97.24	2.66	97.30	25.70	97.28	16.20	97.29	20.49
100	97.00	0.54	97.02	1.31	97.07	2.22	97.08	2.58

Text Chunking with F - <i>dist</i>								
ν	$\omega=1$		$\omega=10$		$\omega=100$		$\omega=\infty$	
	F	time	F	time	F	time	F	time
1	93.95	7.42	94.30	23.30	94.22	34.74	94.31	21.26
10	93.99	0.98	94.08	2.44	94.19	3.11	94.18	3.18
100	93.32	0.16	93.33	0.32	93.42	0.40	93.42	0.40

Text Chunking with W - <i>dist</i>								
ν	$\omega=1$		$\omega=10$		$\omega=100$		$\omega=\infty$	
	F	time	F	time	F	time	F	time
1	93.99	2.93	94.24	24.77	94.32	35.72	94.32	35.61
10	93.98	0.71	94.30	2.82	94.29	3.60	94.30	4.05
100	93.66	0.17	93.65	0.36	93.50	0.42	93.50	0.42

better average accuracy than $\zeta = \{1, 3\}$ in both tasks. Table 4 lists comparison of F -*dist* and W -*dist* on POS tagging and text chunking. Most of accuracy obtained with W -*dist*-based taggers and parsers better than accuracy obtained with F -*dist*-based taggers and parsers. These results have shown that W -*dist* improves accuracy without drastically increasing training time. The text chunker and the tagger trained with AdaBoost.SDF ($\nu = 10$, $\omega = 10$ and W -*dist*) has shown competitive accuracy with that of the chunker trained with AdaBoost.SDF ($\nu = 1$, $\omega = \infty$ and F -*dist*) while maintaining about 7.5 times faster training speed.

5.5 Tagging and Chunking Speeds

We measured testing speeds of taggers and chunkers based on rules or models listed in Table 5.¹⁰

We examined two types of fast classification algorithms for polynomial kernel: Polynomial Kernel Inverted (PKI) and Polynomial Kernel Expanded (PKE). The PKI leads to about 2 to 12 times improvements, and the PKE leads to 30 to 300 compared with normal classification approach of SVM (Kudo and Matsumoto, 2003).¹¹

The POS-taggers based on AdaBoost.SDF, SVM with PKI, and SVM with PKE processed 4,052 words, 159 words, and 1,676 words per second, respectively. The chunkers based on these three methods processed 2,732 words, 113 words, and 1,718 words per second, respectively.

¹⁰We list average speeds of three times tests measured with a machine with Xeon 3.8 GHz CPU and 4 GB of memory.

¹¹We use a chunker YamCha for evaluating classification speeds based on PKI or PKE (<http://www.chasen.org/~taku/software/yamcha/>). We list the average speeds of SVM-based tagger and chunker with PKE of a threshold parameter $\sigma = 0.0005$ for rule selection in both task. The accuracy obtained with models converted by PKE are slightly lower than the accuracy obtained with their original models in our experiments.

Table 5: Comparison with previous best results: (Top : POS tagging, Bottom: Text Chunking)

POS tagging	$F_{\beta=1}$
Perceptron (Collins, 2002)	97.11
Dep. Networks (Toutanova et al., 2003)	97.24
SVM (Giménez and Márquez, 2003)	97.05
ME based a bidirectional inference (Tsuruoka and Tsujii, 2005)	97.15
Guided learning for bidirectional sequence classification (Shen et al., 2007)	97.33
AdaBoost.SDF with candidate features ($\zeta=2, \nu=1, \omega=100, W\text{-dist}$)	97.32
AdaBoost.SDF with candidate features ($\zeta=2, \nu=10, \omega=10, F\text{-dist}$)	97.32
SVM with candidate features ($C=0.1, d=2$)	97.32
Text Chunking	$F_{\beta=1}$
Regularized Winnow + full parser output (Zhang et al., 2001)	94.17
SVM-voting (Kudo and Matsumoto, 2001)	93.91
ASO + unlabeled data (Ando and Zhang, 2005)	94.39
CRF+Reranking(Kudo et al., 2005)	94.12
ME based a bidirectional inference (Tsuruoka and Tsujii, 2005)	93.70
LaSo (Approximate Large Margin Update) (Daumé III and Marcu, 2005)	94.4
HySOL (Suzuki et al., 2007)	94.36
AdaBoost.SDF with candidate features ($\zeta=2, \nu=1, \omega=\infty, W\text{-dist}$)	94.32
AdaBoost.SDF with candidate features ($\zeta=2, \nu=10, \omega=10, W\text{-dist}$)	94.30
SVM with candidate features ($C=1, d=2$)	94.31

One of the reasons that boosting-based classifiers realize faster classification speed is sparseness of rules. SVM learns a final hypothesis as a linear combination of the training examples using some coefficients. In contrast, this boosting-based rule learner learns a final hypothesis that is a subset of candidate rules (Kudo and Matsumoto, 2004).

6 Related Works

6.1 Comparison with Previous Best Results

We list previous best results on English POS tagging and Text chunking in Table 5. These results obtained with the taggers and chunkers based on AdaBoost.SDF and SVM showed competitive F-measure with previous best results. These show that candidate features contribute to create state-of-the-art taggers and chunkers.

These results have also shown that AdaBoost.SDF-based taggers and chunkers show competitive accuracy by learning combination of features automatically. Most of these previous works manually selected combination of features except for SVM with polynomial kernel and (Kudo and Matsumoto, 2001) a boosting-based re-ranking (Kudo et al., 2005).

6.2 Comparison with Boosting-based Learners

LazyBoosting randomly selects a small proportion of features and selects a rule represented by a feature from the selected features at each iteration (Escudero et al., 2000).

Collins and Koo proposed a method only updates values of features co-occurring with a rule feature on examples at each iteration (Collins and Koo, 2005).

Kudo et al. proposed to perform several pseudo iterations for converging fast (Kudo et al., 2005) with features in the cache that maintains the features explored in the previous iterations.

AdaBoost.MH^{KR} learns a weak-hypothesis represented by a set of rules at each boosting iteration

(Sebastiani et al., 2000).

AdaBoost.SDF differs from previous works in the followings. AdaBoost.SDF learns several rules at each boosting iteration like AdaBoost.MH^{KR}. However, the confidence value of each hypothesis in AdaBoost.MH^{KR} does not always minimize the upper bound of training error for AdaBoost because the value of each hypothesis consists of the sum of the confidence value of each rule. Compared with AdaBoost.MH^{KR}, AdaBoost.SDF computes the confidence value of each rule to minimize the upper bound of training error on given weights of samples at each update.

Furthermore, AdaBoost.SDF learns several rules represented by combination of features from limited search spaces at each boosting iteration. The creation of subsets of features in AdaBoost.SDF enables us to recreate the same classifier with same parameters and training data. Recreation is not ensured in the random selection of subsets in LazyBoosting.

7 Conclusion

We have proposed a fast boosting-based learner, which we call AdaBoost.SDF. AdaBoost.SDF repeats to learn several rules represented by combination of features from a small proportion of candidate rules. We have also proposed methods to use candidate POS tags and chunk tags of each word obtained from automatically tagged data as features in POS tagging and text chunking.

The experimental results have shown drastically improvement of training speed while maintaining competitive accuracy compared with previous best results.

Future work should examine our approach on several tasks. Future work should also compare our algorithm with other learning algorithms.

Appendix A: Convergence

The upper bound of the training error for AdaBoost of (Freund and Mason, 1999), which is used in AdaBoost.SDF, is induced by adopting THEOREM 1 presented in (Schapire and Singer, 1999). Let Z_R be $\sum_{i=1}^m w_{R+1,i}$ that is a sum of weights updated with R rules. The bound holds on the training error after selecting R rules,

$$\sum_{i=1}^m [[F(\mathbf{x}_i) \neq y_i]] \leq Z_R$$

is induced as follows.

By unraveling the Eq. (1), we obtain $w_{R+1,i} = \exp(-y_i \sum_{r=1}^R h_{(\mathbf{f}_r, c_r)}(\mathbf{x}_i))$. Thus, we obtain $[[F(\mathbf{x}_i) \neq y_i]] \leq \exp(-y_i \sum_{t=1}^R h_{(\mathbf{f}_t, c_t)}(\mathbf{x}_i))$, since if $F(\mathbf{x}_i) \neq y_i$, then $\exp(-y_i \sum_{r=1}^R h_{(\mathbf{f}_r, c_r)}(\mathbf{x}_i)) \geq 1$. Combining these equations gives the stated bound on training error

$$\begin{aligned}
\sum_{i=1}^m [[F(\mathbf{x}_i) \neq y_i]] &\leq \sum_{i=1}^m \exp(-y_i \sum_{t=1}^R h_{(\mathbf{f}_t, c_t)}(\mathbf{x}_i)) \\
&= \sum_{i=1}^m w_{R+1,i} = Z_R. \quad (2)
\end{aligned}$$

Then we show that the upper bound of training error Z_R for R rules shown in Eq. (2) is less than or equal to the upper bound of the training error Z_{R-1} for $R-1$ rules. By unraveling the (2) and plugging the confidence values $c_R = \{\frac{1}{2} \log(\frac{W_{r,+1}(\mathbf{f}_R)}{W_{r,-1}(\mathbf{f}_R)}), 0\}$ given by the weak hypothesis into the unraveled equation, we obtain $Z_R \leq Z_{R-1}$, since

$$\begin{aligned}
Z_R &= \sum_{i=1}^m w_{R+1,i} = \sum_{i=1}^m w_{R,i} \exp(-y_i h_{(\mathbf{f}_R, c_R)}) \\
&= \sum_{i=1}^m w_{R,i} - W_{r,+1}(\mathbf{f}_R) - W_{r,+1}(\mathbf{f}_R) + \\
&\quad W_{r,+1}(\mathbf{f}_R) \exp(-c_R) + W_{r,-1}(\mathbf{f}_R) \exp(c_R) \\
&= Z_{R-1} - (\sqrt{W_{r,+1}(\mathbf{f}_R)} - \sqrt{W_{r,-1}(\mathbf{f}_R)})^2
\end{aligned}$$

References

- Ando, Rie and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proc. of 43rd Meeting of Association for Computational Linguistics*, pages 1–9.
- Collins, Michael and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.
- Collins, Michael. 2002. Discriminative training methods for Hidden Markov Models: theory and experiments with perceptron algorithms. In *Proc. of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8.
- Daumé III, Hal and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proc. of 22th International Conference on Machine Learning*, pages 169–176.
- Escudero, Gerard, Lluís Màrquez, and German Rigau. 2000. Boosting applied to word sense disambiguation. In *Proc. of 11th European Conference on Machine Learning*, pages 129–141.
- Freund, Yoav and Llew Mason. 1999. The alternating decision tree learning algorithm. In *Proc. of 16th International Conference on Machine Learning*, pages 124–133.
- Giménez, Jesús and Lluís Màrquez. 2003. Fast and accurate part-of-speech tagging: The SVM approach revisited. In *Proc. of International Conference Recent Advances in Natural Language Processing 2003*, pages 153–163.
- Iwakura, Tomoya and Seishi Okamoto. 2007. Fast training methods of boosting algorithms for text analysis. In *Proc. of International Conference Recent Advances in Natural Language Processing 2007*, pages 274–279.
- Kazama, Jun’ichi and Kentaro Torisawa. 2007. A new perceptron algorithm for sequence labeling with non-local features. In *Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 315–324.
- Kudo, Taku and Yuji Matsumoto. 2001. Chunking with Support Vector Machines. In *Proc. of The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 192–199.
- Kudo, Taku and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proc. of 41st Meeting of Association for Computational Linguistics*, pages 24–31.
- Kudo, Taku and Yuji Matsumoto. 2004. A boosting algorithm for classification of semi-structured text. In *Proc. of the 2004 Conference on Empirical Methods in Natural Language Processing 2004*, pages 301–308, July.
- Kudo, Taku, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proc. of 43rd Meeting of Association for Computational Linguistics*, pages 189–196.
- Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of english: The Penn Treebank. pages 313–330.
- Morishita, Shinichi. 2002. Computing optimal hypotheses efficiently for boosting. *Proc. of 5th International Conference Discovery Science*, pages 471–481.
- Schapire, Robert E. and Yoram Singer. 1999. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336.
- Schapire, Robert E. and Yoram Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.
- Sebastiani, Fabrizio, Alessandro Sperduti, and Nicola Valdambrini. 2000. An improved boosting algorithm and its application to text categorization. In *Proc. of International Conference on Information and Knowledge Management*, pages 78–85.
- Shen, Libin, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proc. of 45th Meeting of Association for Computational Linguistics*, pages 760–767.
- Suzuki, Jun, Akinori Fujino, and Hideki Isozaki. 2007. Semi-supervised structured output learning based on a hybrid generative and discriminative approach. In *Proc. of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 791–800.
- Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 173–180.
- Tsuruoka, Yoshimasa and Junichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proc. of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 467–474.
- Zhang, Tong, Fred Damerau, and David Johnson. 2001. Text chunking using regularized winnow. In *Proc. of 39th Meeting of Association for Computational Linguistics*, pages 539–546.

Linguistic features in data-driven dependency parsing

Lilja Øvrelid

NLP-unit, Dept. of Swedish
University of Gothenburg
Sweden

`lilja.ovrelid@svenska.gu.se`

Abstract

This article investigates the effect of a set of linguistically motivated features on argument disambiguation in data-driven dependency parsing of Swedish. We present results from experiments with gold standard features, such as animacy, definiteness and finiteness, as well as corresponding experiments where these features have been acquired automatically and show significant improvements both in overall parse results and in the analysis of specific argument relations, such as subjects, objects and predicatives.

1 Introduction

Data-driven dependency parsing has recently received extensive attention in the parsing community and impressive results have been obtained for a range of languages (Nivre et al., 2007). Even with high overall parsing accuracy, however, data-driven parsers often make errors in the assignment of argument relations such as subject and object and the exact influence of data-derived features on the parsing accuracy for specific linguistic constructions is still relatively poorly understood. There are a number of studies that investigate the influence of different features or representational choices on overall parsing accuracy, (Bod, 1998; Klein and Manning, 2003). There are also attempts at a more fine-grained analysis of accuracy, targeting specific linguistic constructions or grammatical functions (Carroll and Briscoe, 2002; Kübler and Prokić, 2006; McDonald and Nivre, 2007).

©2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

But there are few studies that combine the two perspectives and try to tease apart the influence of different features on the analysis of specific constructions, let alone motivated by a thorough linguistic analysis.

In this paper, we investigate the influence of a set of linguistically motivated features on parse results for Swedish, and in particular on the analysis of argument relations such as subjects, objects and subject predicatives. Motivated by an error analysis of the best performing parser for Swedish in the CoNLL-X shared task, we extend the feature model employed by the parser with a set of linguistically motivated features and go on to show how these features may be acquired automatically. We then present results from corresponding parse experiments with automatic features.

The rest of the paper is structured as follows. In section 2 we present relevant properties of Swedish morphosyntax, as well as the treebank and parser employed in the experiments. Section 3 presents an error analysis of the baseline parser and we go on to motivate a set of linguistic features in section 4, which are employed in a set of experiments with gold standard features, discussed in section 5. Section 6 presents the automatic acquisition of these features, with a particular focus on animacy classification and in section 7 we report parse experiments with automatic features.

2 Parsing Swedish

Before we turn to a description of the treebank and the parser used in the experiments, we want to point to a few grammatical properties of Swedish that will be important in the following:

Verb second (V2) Swedish is, like the majority of Germanic languages a V2-language; the finite verb always resides in second position in

declarative main clauses.

Word order variation Pretty much any constituent may occupy the sentence-initial position, but subjects are most common.

Limited case marking Nouns are only inflected for genitive case. Personal pronouns distinguish nominative and accusative case, but demonstratives and quantifying pronouns are case ambiguous (like nouns).

2.1 Treebank: Talbanken05

Talbanken05 is a Swedish treebank converted to dependency format, containing both written and spoken language (Nivre et al., 2006a).¹ For each token, Talbanken05 contains information on word form, part of speech, head and dependency relation, as well as various morphosyntactic and/or lexical semantic features. The nature of this additional information varies depending on part of speech:

NOUN:	<i>definiteness, animacy, case (Ø/GEN)</i>
PRO:	<i>animacy, case (Ø/ACC)</i>
VERB:	<i>tense, voice (Ø/PA)</i>

2.2 Parser: MaltParser

We use the freely available MaltParser,² which is a language-independent system for data-driven dependency parsing. MaltParser is based on a deterministic parsing strategy, first proposed by Nivre (2003), in combination with treebank-induced classifiers for predicting the next parsing action. Classifiers can be trained using any machine learning approach, but the best results have so far been obtained with support vector machines, using LIBSVM (Chang and Lin, 2001). MaltParser has a wide range of parameters that need to be optimized when parsing a new language. As our baseline, we use the settings optimized for Swedish in the CoNLL-X shared task (Nivre et al., 2006b), where this parser was the best performing parser for Swedish. The only parameter that will be varied in the later experiments is the feature model used for the prediction of the next parsing action. Hence, we need to describe the feature model in a little more detail.

MaltParser uses two main data structures, a stack (S) and an input queue (I), and builds a dependency graph (G) incrementally in a single left-

¹The written sections of the treebank consist of professional prose and student essays and amount to 197,123 running tokens, spread over 11,431 sentences.

²<http://w3.msi.vxu.se/users/nivre/research/MaltParser.html>

	FORM	POS	DEP	FEATS
S:top	+	+	+	+
S:top+1				
I:next	+	+		+
I:next-1	+			+
I:next+1	+	+		+
I:next+2		+		
G: head of top	+			+
G: left dep of top			+	
G: right dep of top			+	
G: left dep of next	+		+	+
G: left dep of head of top			+	
G: left sibling of right dep of top			+	
G: right sibling of left dep of top	+			+
G: right sibling of left dep of next		+	+	

Table 1: Baseline and extended (FEATS) feature model for Swedish; S: stack, I: input, G: graph; $\pm n = n$ positions to the left(-) or right(+)

to-right pass over the input. The decision that needs to be made at any point during this derivation is (a) whether to add a dependency arc (with some label) between the token on top of the stack (*top*) and the next token in the input queue (*next*), and (b) whether to pop *top* from the stack or push *next* onto the stack. The features fed to the classifier for making these decisions naturally focus on attributes of *top*, *next* and neighbouring tokens in S, I or G. In the baseline feature model, these attributes are limited to the word form (FORM), part of speech (POS), and dependency relation (DEP) of a given token, but in later experiments we will add other linguistic features (FEATS). The baseline feature model is depicted as a matrix in Table 1, where rows denote tokens in the parser configuration (defined relative to S, I and G) and columns denote attributes. Each cell containing a + corresponds to a feature of the model.

3 Baseline and Error Analysis

The written part of Talbanken05 was parsed employing the baseline feature model detailed above, using 10-fold cross validation for training and testing. The overall result for unlabeled and labeled dependency accuracy is 89.87 and 84.92 respectively.³

Error analysis shows that the overall most frequent errors in terms of dependency relations involve either various adverbial relations, due to PP-attachment ambiguities and a large number of ad-

³Note that these results are slightly better than the official CoNLL-X shared task scores (89.50/84.58), which were obtained using a single training-test split, not cross-validation. Note also that, in both cases, the parser input contained gold standard part-of-speech tags.

Gold	Sys	before	after	Total
ss	oo	103 (23.1%)	343 (76.9%)	446 (100%)
oo	ss	103 (33.3%)	206 (66.7%)	309 (100%)

Table 2: Position relative to verb for confused subjects and objects

verbal labels, or the argument relations, such as subjects, direct objects, formal subjects and subject predicatives. In particular, confusion of argument relations are among the most frequent error types with respect to dependency assignment.⁴

Swedish exhibits some ambiguities in word order and morphology which follow from the properties discussed above. We will exemplify these factors through an analysis of the errors where subjects are assigned object status (SS_OO) and vice versa (OO_SS). The confusion of subjects and objects follows from lack of sufficient formal disambiguation, i.e., simple clues such as word order, part-of-speech and word form do not clearly indicate syntactic function.

With respect to word order, subjects and objects may both precede or follow their verbal head. Subjects, however, are more likely to occur preverbally (77%), whereas objects typically occupy a postverbal position (94%). We would therefore expect postverbal subjects and preverbal objects to be more dominant among the errors than in the treebank as a whole (23% and 6% respectively). Table 2 shows a breakdown of the errors for confused subjects and objects and their position with respect to the verbal head. We find that postverbal subjects (after) are in clear majority among the subjects erroneously assigned the object relation. Due to the V2 property of Swedish, the subject must reside in the position directly following the finite verb whenever another constituent occupies the preverbal position, as in (1) where a direct object resides sentence-initially:

- (1) *Samma erfarenhet gjorde engelsmännen*
 same experience made englishmen-DEF
 ‘The same experience, the Englishmen had’

For the confused objects we find a larger proportion of preverbal elements than for subjects, which

⁴We define argument relations as dependency relations which obtain between a verb and a dependent which is subcategorized for and/or thematically entailed by the verb. Note that arguments are not distinguished structurally from non-arguments, like adverbials, in dependency grammar, but through dependency label.

is the mirror image of the normal distribution of syntactic functions among preverbal elements. As Table 2 shows, the proportion of preverbal elements among the subject-assigned objects (33.3%) is notably higher than in the corpus as a whole, where preverbal objects account for a miniscule 6% of all objects.

In addition to the word order variation discussed above, Swedish also has limited morphological marking of syntactic function. Nouns are marked only for genitive case and only pronouns are marked for accusative case. There is also syncretism in the pronominal paradigm where the pronoun is invariant for case, e.g. *det, den* ‘it’, *ingen/inga* ‘no’, and may, in fact, also function as a determiner. This means that, with respect to word form, only the set of unambiguous pronouns clearly indicate syntactic function. In the errors, we find that nouns and functionally ambiguous pronouns dominate the errors where subjects and objects are confused, accounting for 84.5% of the SS_OO and 93.5% of the OO_SS errors.

The initial error analysis shows that the confusion of argument relations constitutes a frequent and consistent error during parsing. Ambiguities in word order and morphological marking constitute a complicating factor and we find cases that deviate from the most frequent word order patterns and are not formally disambiguated by part-of-speech information. It is clear that we in order to resolve these ambiguities have to examine features beyond syntactic category and linear word order.

4 Linguistic features for argument disambiguation

Argument relations tend to differ along several linguistic dimensions. These differences are found as statistical tendencies, rather than absolute requirements on syntactic structure. The property of *animacy*, a referential property of nominal elements, has been argued to play a role in argument realization in a range of languages see de Swart et.al. (2008) for an overview. It is closely correlated with the semantic property of agentivity, hence subjects will tend to be referentially animate more often than objects. Another property which may differentiate between the argument functions is the property of *definiteness*, which can be linked with a notion of givenness, (Weber and Müller, 2004). This is reflected in the choice of referring expression for the various argument types in

Talbanken05 – subjects are more often pronominal (49.2%), whereas objects and subject predicatives are typically realized by an indefinite noun (67.6% and 89.6%, respectively). As mentioned in section 2, there are categorical constraints which are characteristic for Swedish morphosyntax. Even if the morphological marking of arguments in Scandinavian is not extensive or unambiguous, *case* may distinguish arguments. Only subjects may follow a finite verb and precede a non-finite verb and only complements may follow a non-finite verb. Information on *tense* or the related *finiteness* is therefore something that one might assume to be beneficial for argument analysis. Another property of the verb which clearly influences the assignment of core argument functions is the *voice* of the verb, i.e., whether it is passive or active.⁵

5 Experiments with gold standard features

We perform a set of experiments with an extended feature model and added, gold standard information on animacy, definiteness, case, finiteness and voice, where the features were employed individually as well as in combination.

5.1 Experimental methodology

All parsing experiments are performed using 10-fold cross-validation for training and testing on the entire written part of Talbanken05. The feature model used throughout is the extended feature model depicted in Table 1, including all four columns.⁶ Hence, what is varied in the experiments is only the information contained in the FEATS features (animacy, definiteness, etc.), while the tokens for which these features are defined remains constant. Overall parsing accuracy will be reported using the standard metrics of *labeled attachment score* (LAS) and *unlabeled attachment score* (UAS).⁷ Statistical significance is checked using Dan Bikel’s randomized parsing evaluation

⁵We experimented with the use of tense as well as finiteness, a binary feature which was obtained by a mapping from tense to finite/non-finite. Finiteness gave significantly better results ($p < .03$) and was therefore employed in the following, see (Øvrelid, 2008b) for details.

⁶Preliminary experiments showed that it was better to tie FEATS features to the same tokens as FORM features (rather than POS or DEP features). Backward selection from this model was tried for several different instantiations of FEATS but with no significant improvement.

⁷LAS and UAS report the percentage of tokens that are assigned the correct head *with* (labeled) or *without* (unlabeled) the correct dependency label, calculated using eval.pl with default settings (<http://nextens.uvt.nl/~conll/software.html>)

comparator.⁸ Since the main focus of this article is on the disambiguation of grammatical functions, we report accuracy for specific dependency relations, measured as a balanced F-score.

5.2 Results

The overall results for these experiments are presented in table 3, along with p-scores. The experiments show that each feature individually causes a significant improvement in terms of overall labeled accuracy as well as performance for argument relations. Error analysis comparing the baseline parser (NoFeats) with new parsers trained with individual features reveal the influence of these features on argument disambiguation. We find that animacy influences the disambiguation of subjects from objects, objects from indirect objects as well as the general distinction of arguments from non-arguments. Definiteness has a notable effect on the disambiguation of subjects and subject predicatives. Information on morphological case shows a clear effect in distinguishing between arguments and non-arguments, and in particular, in distinguishing nominal modifiers with genitive case. The added verbal features, finiteness and voice, have a positive effect on the verbal dependency relations, as well as an overall effect on the assignment of the SS and OO argument relations. Information on voice also benefits the relation expressing the demoted agent (AG) in passive constructions, headed by the preposition *av* ‘by’, as in English.

The ADCV experiment which combines information on animacy, definiteness, case and verbal features shows a cumulative effect of the added features with results which differ significantly from the baseline, as well as from each of the individual experiments ($p < .0001$). We observe clear improvements for the analysis of all argument relations, as shown by the third column in table 4 which presents F-scores for the various argument relations.

6 Acquiring features

A possible objection to the general applicability of the results presented above is that the added information consists of gold standard annotation from a treebank. However, the morphosyntactic features examined here (definiteness, case, tense, voice) represent standard output from most part-of-speech taggers. In the following we will also

⁸<http://www.cis.upenn.edu/~dbikel/software.html>

	UAS	LAS	p-value
NoFeats	89.87	84.92	–
Anim	89.93	85.10	p<.0002
Def	89.87	85.02	p<.02
Case	89.99	85.13	p<.0001
Verb	90.24	85.38	p<.0001
ADC	90.13	85.35	p<.0001
ADCV	90.40	85.68	p<.0001

Table 3: Overall results in gold standard experiments expressed as unlabeled and labeled attachment scores.

Feature	Application
Definiteness	POS-tagger
Case	POS-tagger
Animacy - NN	Animacy classifier
Animacy - PN	Named Entity Tagger
Animacy - PO	Majority class
Tense (finiteness), voice	POS-tagger

Table 5: Overview of applications employed for automatic feature acquisition.

show that the property of animacy can be fairly robustly acquired for common nouns by means of distributional features from an automatically parsed corpus.

Table 5 shows an overview of the applications employed for the automatic acquisition of our linguistic features. For part-of-speech tagging, we employ MaltTagger – a HMM part-of-speech tagger for Swedish (Hall, 2003). The POS-tagger distinguishes tense and voice for verbs, nominative and accusative case for pronouns, as well as definiteness and genitive case for nouns.

6.1 Animacy

The feature of animacy is clearly the most challenging feature to acquire automatically. Recall that Talbanken05 distinguishes animacy for all nominal constituents. In the following we describe the automatic acquisition of animacy information for common nouns, proper nouns and pronouns.

Common nouns Table 6 presents an overview of the animacy data for common nouns in Talbanken05. It is clear that the data is highly skewed

	NoFeats	Gold	Auto
SS subject	90.25	91.80	91.32
OO object	84.53	86.27	86.10
SP subj.pred.	84.82	85.87	85.80
AG pass. agent	73.56	81.34	81.02
ES logical subj.	71.82	73.44	72.60
FO formal obj.	56.68	65.64	65.38
VO obj. small clause	72.10	83.40	83.12
VS subj. small clause	58.75	65.56	68.75
FS formal subj.	71.31	72.10	71.31
IO indir. obj.	76.14	77.76	76.29

Table 4: F-scores for argument relations with combined features (ADCV).

Class	Types	Tokens covered
Animate	644	6010
Inanimate	6910	34822
Total	7554	40832

Table 6: The animacy data set from Talbanken05; number of noun lemmas (Types) and tokens in each class.

towards the non-person class, which accounts for 91.5% of the data instances. Due to the small size of the treebank we classify common noun *lemmas* based on their morphosyntactic distribution in a considerably larger corpus. For the animacy classification of common nouns, we construct a general *feature space* for animacy classification, which makes use of distributional data regarding syntactic properties of the noun, as well as various morphological properties. The syntactic and morphological features in the general feature space are presented below:

Syntactic features A feature for each dependency relation with nominal potential: (transitive) subject (SUBJ), object (OBJ), prepositional complement (PA), root (ROOT)⁹, apposition (APP), conjunct (CC), determiner (DET), predicative (PRD), complement of comparative subjunction (UK). We also include a feature for the complement of a genitive modifier, the so-called ‘possessee’, (GENHD).

Morphological features A feature for each mor-

⁹Nominal elements may be assigned the root relation in sentence fragments which do not include a finite verb.

phological distinction relevant for a noun: gender (NEU/UTR), number (SIN/PLU), definiteness (DEF/IND), case (NOM/GEN). Also, the part-of-speech tags distinguish dates (DAT) and quantifying nouns (SET), e.g. *del*, *rad* ‘part, row’, so these are also included as features.

For extraction of distributional data for the Talbanken05 nouns we make use of the Swedish Parole corpus of 21.5M tokens.¹⁰ To facilitate feature extraction, we part-of-speech tag the corpus and parse it with MaltParser, which assigns a dependency analysis.¹¹ For classification, we make use of the Tilburg Memory-Based Learner (TiMBL) (Daelemans et al., 2004).¹² and optimize the TiMBL parameters on a subset of the full data set.¹³

We obtain results for animacy classification of noun lemmas, ranging from 97.3% accuracy to 94.0% depending on the sparsity of the data. With an absolute frequency threshold of 10, we obtain an accuracy of 95.4%, which constitutes a 50% reduction of error rate over a majority baseline. We find that classification of the inanimate class is quite stable throughout the experiments, whereas the classification of the minority class of animate nouns suffers from sparse data. We obtain a F-score of 71.8% F-score for the animate class and 97.5% for the inanimate class with a threshold of 10. The common nouns in Talbanken05 are classified for animacy following a leave-one-out training and testing scheme where each of the n nouns in Talbanken05 are classified with a classifier trained on $n - 1$ instances. This ensures that the training and test instances are disjoint at all times. Moreover, the fact that the distributional data is taken from a separate data set ensures non-circularity

¹⁰Parole is available at <http://spraakbanken.gu.se>

¹¹For part-of-speech tagging, we employ the MaltTagger – a HMM part-of-speech tagger for Swedish (Hall, 2003). For parsing, we employ MaltParser with a pretrained model for Swedish, which has been trained on the tags output by the tagger. It makes use of a smaller set of dependency relations than those found in Talbanken05.

¹²TiMBL is freely available at <http://ilk.uvt.nl/software.html>

¹³For parameter optimization we employ the paramsearch tool, supplied with TiMBL, see <http://ilk.uvt.nl/software.html>. Paramsearch implements a hill climbing search for the optimal settings on iteratively larger parts of the supplied data. We performed parameter optimization on 20% of the total >0 data set, where we balanced the data with respect to frequency. The resulting settings are $k = 11$, GainRatio feature weighting and Inverse Linear (IL) class voting weights.

since we are not basing the classification on gold standard parses.

Proper nouns In the task of named entity recognition (NER), proper nouns are classified according to a set of semantic categories. For the annotation of proper nouns, we make use of a named entity tagger for Swedish (Kokkinakis, 2004), which is a rule-based tagger based on finite-state rules, supplied with name lists, so-called “gazetteers”. The tagger distinguishes the category ‘Person’ for human referring proper nouns and we extract information on this category.

Pronouns A subset of the personal pronouns in Scandinavian, as in English, clearly distinguish their referent with regard to animacy, e.g. *han*, *det* ‘he, it’. There is, however, a quite large group of third person plural pronouns which are ambiguous with regards to the animacy of their referent, e.g., *de*, *dem*, *deras* ‘they, them, theirs’. Pronominal reference resolution is a complex task which we will not attempt to solve in the present context. The pronominal part-of-speech tags from the part-of-speech tagger distinguish number and gender and in the animacy classification of the personal pronouns we classify based on these tags only. We employ a simple heuristic where the pronominal tags which had more than 85% human instances in the gold standard are annotated as human.¹⁴ The pronouns which are ambiguous with respect to animacy are not annotated as animate.

In table 7 we see an overview of the accuracy of the acquired features, i.e., the percentage of correct instances out of all instances. Note that we adhere to the general annotation strategy in Talbanken05, where each dimension (definiteness, case etc.) contains a null category \emptyset , which expresses the lack of a certain property. The acquisition of the morphological features (definiteness, case, finiteness and voice) are very reliable, with accuracies from 96.9% for voice to 98.5% for the case feature.

It is not surprising that we observe the largest discrepancies from the gold standard annotation in the automatic animacy annotation. In general, the annotation of animate nominals exhibits a decent precision (95.7) and a lower recall (61.3). The automatic classification of human common nouns

¹⁴A manual classification of the individual pronoun lemmas was also considered. However, the treebank has a total of 324 different pronoun forms, hence we opted for a heuristic classification of the part-of-speech tags instead.

Dimension	Features	Instances	Correct	Accuracy
Definiteness	DD, \emptyset	40832	40010	98.0
Case	GG, AA, \emptyset	68313	67289	98.5
Animacy _{NNPNPO}	HH, \emptyset	68313	61295	89.7
Animacy _{NN}	HH, \emptyset	40832	37952	92.9
Animacy _{PN}	HH, \emptyset	2078	1902	91.5
Animacy _{PO}	HH, \emptyset	25403	21441	84.4
Finiteness	FV, \emptyset	30767	30035	97.6
Voice	PA, \emptyset	30767	29805	96.9

Table 7: Accuracy for automatically acquired linguistic features.

	Gold		Automatic		p-value
	UAS	LAS	UAS	LAS	
NoFeats	89.87	84.92	89.87	84.92	–
Def	89.87	85.02	89.88	85.03	$p < 0.01$
Case	89.99	85.13	89.95	85.11	$p < .0001$
Verb	90.24	85.38	90.12	85.26	$p < .0001$
Anim	89.93	85.10	89.86	85.01	$p < .03$
ADC	90.13	85.35	90.01	85.21	$p < .0001$
ADCV	90.40	85.68	90.27	85.54	$p < .0001$

Table 8: Overall results in experiments with automatic features compared to gold standard features.

(Animacy_{NN}) also has a quite high precision (94.2) in combination with a lower recall (55.5). The named-entity recognizer (Animacy_{PN}) shows more balanced results with a precision of 97.8 and a recall of 85.2 and the heuristic classification of the pronominal part-of-speech tags (Animacy_{PO}) gives us high precision (96.3) combined with lower recall (62.0) for the animate class.

7 Experiments with acquired features

The experimental methodology is identical to the one described in 5.1 above, the only difference being that the linguistic features are acquired automatically, rather than being gold standard. In order to enable a direct comparison with the results from the earlier experiments, we employ the gold standard part-of-speech tags, as before. This means that the set for which the various linguistic features are defined is identical, whereas the feature values may differ.

Table 8 presents the overall results with automatic features, compared to the gold standard results and p-scores for the difference of the automatic results from the NoFeats baseline. As ex-

pected, we find that the effect of the automatic features is generally lower than their gold standard counterparts. However, all automatic features improve significantly on the NoFeats baseline. In the error analysis we find the same tendencies in terms of improvement for specific dependency relations.

The morphological argument features from the POS-tagger are reliable, as we saw above, and we observe almost identical results to the gold standard results. The addition of information on definiteness causes a significant improvement ($p < .01$), and so does the addition of information on case ($p < .0001$). The addition of the automatically acquired animacy information results in a smaller, but significant improvement of overall results even though the annotation is less reliable ($p < .03$). An interesting result is that the automatically acquired information on animacy for common nouns actually has a significantly better effect than the gold standard counterparts due to capturing distributional tendencies (Øvrelid, 2008a). As in the gold standard experiments, we find that the features which have the most notable effect on performance are the verbal features ($p < .0001$).

In parallel with the results achieved with the combination of gold standard features, we observe improvement of overall results compared to the baseline ($p < .0001$) and each of the individual features when we combine the features of the arguments (ADC; $p < .01$) and the argument and verbal features (ADCV; $p < .0001$). Column 4 in Table 4 shows an overview of performance for the argument relations, compared to the gold standard experiments. We find overall somewhat lower results in the experiment with automatic features, but find the same tendencies with the automatically acquired features.

8 Conclusion

An error analysis of the best performing data-driven dependency parser for Swedish revealed consistent errors in dependency assignment, namely the confusion of argument functions. We established a set of features expressing distinguishing semantic and structural properties of arguments such as animacy, definiteness and finiteness and performed a set of experiments with gold standard features taken from a treebank of Swedish. The experiments showed that each feature individually caused an improvement in terms of overall labeled accuracy and performance for the argument relations. We furthermore found that the results may largely be replicated with automatic features and a generic part-of-speech tagger. The features were acquired automatically employing a part-of-speech tagger, a named-entity recognizer and an animacy classifier of common noun lemmas employing morphosyntactic distributional features. A set of corresponding experiments with automatic features gave significant improvement from the addition of individual features and a cumulative effect of the same features in combination. In particular, we show that the very same tendencies in improvement for specific argument relations such as subjects, objects and predicatives may be obtained using automatically acquired features.

Properties of the Scandinavian languages connected with errors in argument assignment are not isolated phenomena. A range of other languages exhibit similar properties, for instance, Italian exhibits word order variation, little case, syncretism in agreement morphology, as well as pro-drop; German exhibits a larger degree of word order variation in combination with quite a bit of syncretism in case morphology; Dutch has word order variation, little case and syncretism in agreement morphology. These are all examples of other languages for which the results described here are relevant.

References

- Bod, Rens. 1998. *Beyond Grammar: An experience-based theory of language*. CSLI Publications, Stanford, CA.
- Carroll, John and Edward Briscoe. 2002. High precision extraction of grammatical relations. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, pages 134–140.
- Chang, Chih-Chung and Chih-Jen Lin. 2001. LIBSVM: A library for support vector machines. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- Daelemans, Walter, Jakub Zavrel, Ko Van der Sloot, and Antal Van den Bosch. 2004. TiMBL: Tilburg Memory Based Learner, version 5.1, Reference Guide. Technical report, ILK Technical Report Series 04-02.
- de Swart, Peter, Monique Lamers, and Sander Lestrade. 2008. Animacy, argument structure and argument encoding: Introduction to the special issue on animacy. *Lingua*, 118(2):131–140.
- Hall, Johan. 2003. A probabilistic part-of-speech tagger with suffix probabilities. Master's thesis, Växjö University, Sweden.
- Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 423–430.
- Kokkinakis, Dimitrios. 2004. Reducing the effect of name explosion. In *Proceedings of the LREC Workshop: Beyond Named Entity Recognition, Semantic labelling for NLP tasks*.
- Kübler, Sandra and Jelena Prokić. 2006. Why is German dependency parsing more reliable than constituent parsing? In *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories (TLT)*, pages 7–18.
- McDonald, Ryan and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing. In *Proceedings of the Eleventh Conference on Computational Natural Language Learning (CoNLL)*, pages 122–131.
- Nivre, Joakim, Jens Nilsson, and Johan Hall. 2006a. Talbanken05: A Swedish treebank with phrase structure and dependency annotation. In *Proceedings of the fifth International Conference on Language Resources and Evaluation (LREC)*, pages 1392–1395.
- Nivre, Joakim, Jens Nilsson, Johan Hall, Gülşen Eryiğit, and Svetoslav Marinov. 2006b. Labeled pseudo-projective dependency parsing with Support Vector Machines. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.
- Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. CoNLL 2007 Shared Task on Dependency Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.
- Øvrelid, Lilja. 2008a. *Argument Differentiation. Soft constraints and data-driven models*. Ph.D. thesis, University of Gothenburg.
- Øvrelid, Lilja. 2008b. Finite matters: Verbal features in data-driven parsing of Swedish. In *Proceedings of the International Conference on NLP, GoTAL 2008*.
- Weber, Andrea and Karin Müller. 2004. Word order variation in German main clauses: A corpus analysis. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 71–77.

Transforming Meaning Representation Grammars to Improve Semantic Parsing

Rohit J. Kate

Department of Computer Sciences*
The University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233, USA
rjkate@cs.utexas.edu

Abstract

A semantic parser learning system learns to map natural language sentences into their domain-specific formal meaning representations, but if the constructs of the meaning representation language do not correspond well with the natural language then the system may not learn a good semantic parser. This paper presents approaches for automatically transforming a meaning representation grammar (MRG) to conform it better with the natural language semantics. It introduces grammar transformation operators and meaning representation macros which are applied in an error-driven manner to transform an MRG while training a semantic parser learning system. Experimental results show that the automatically transformed MRGs lead to better learned semantic parsers which perform comparable to the semantic parsers learned using manually engineered MRGs.

1 Introduction

Semantic parsing is the task of converting *natural language* (NL) sentences into their *meaning representations* (MRs) which a computer program can execute to perform some domain-specific task, like controlling a robot, answering database queries etc. These MRs are expressed in a formal *meaning representation language* (MRL) unique to the domain to suit the application, like some specific command language to control a robot or some

query language to execute database queries. A machine learning system for semantic parsing takes NL sentences paired with their respective MRs as training data and induces a semantic parser which can then map novel NL sentences into their MRs.

The grammar of an MRL, which we will call *meaning representation grammar* (MRG), is assumed to be deterministic and context-free which is true for grammars of almost all the computer executable languages. A semantic parsing learning system typically exploits the given MRG of the MRL to learn a semantic parser (Kate and Mooney, 2006; Wong and Mooney, 2006). Although in different ways, but the systems presented in these papers learn how the NL phrases relate to the *productions* of the MRG, and using this information they parse a test sentence to compositionally generate its best MR. In order to learn a good semantic parser, it is necessary that the productions of the MRG accurately represent the semantics being expressed by the natural language. However, an MRL and its MRG are typically designed to best suit the application with little consideration for how well they correspond to the semantics of a natural language.

Some other semantic parser learning systems which need MRL in the form of Prolog (Tang and Mooney, 2001) or λ -calculus (Zettlemoyer and Collins, 2007; Wong and Mooney, 2007) do not use productions of the MRG but instead use predicates of the MRL. However, in order to learn a good semantic parser, they still require that these predicates correspond well with the semantics of the natural language. There are also systems which learn semantic parsers from more detailed training data in the form of semantically augmented parse trees of NL sentences in which each internal node has a syntactic and a semantic label (Ge

*Alumnus at the time of submission.

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

- (a) NL: *If the ball is in our midfield then player 5 should go to (-5,0).*
 MR: (bpos (rec (pt -32 -35) (pt 0 35))
 (do (player our {5}) (pos (pt -5 0))))
- (b) NL: *Which is the longest river in Texas?*
 MR: answer(longest(river(loc_2(stateid('Texas')))))
- (c) NL: *Which is the longest river in Texas?*
 MR: select river.name from river where
 river.traverse='Texas' and river.length=
 (select max(river.length) from river
 where river.traverse='Texas');

Figure 1: Examples of NL sentences and their MRs from (a) the CLANG domain (b) GEOQUERY domain with functional MRL (c) GEOQUERY domain with SQL.

and Mooney, 2005; Nguyen et al., 2006). For these systems to work well, it is also necessary that the semantic labels of the MRL correspond well with natural language semantics.

If the MRG of a domain-specific MRL does not correspond well with natural language semantics then manually re-engineering the MRG to work well for semantic parsing is a tedious task and requires considerable domain expertise. In this paper, we present methods to automatically transform a given MRG to make it more suitable for learning semantic parsers. No previous work addresses this issue to our best knowledge. We introduce grammar transformation operators and meaning representation macros to transform an MRG. We describe how these are applied in an error-driven manner using the base semantic parsing learning algorithm presented in (Kate and Mooney, 2006) resulting in a better learned semantic parser. Our approach, however, is general enough to improve any semantic parser learning system which uses productions of the MRG. We present experimental results with three very different MRLs to show how these grammar transformations improve the semantic parsing performance.

2 Background

The following subsection gives some examples of semantic parsing domains and their corresponding MRLs and illustrates why incompatibility between MRGs and natural language could hurt semantic parsing. The next subsection then briefly describes a base semantic parser learning system which we use in our experiments.

2.1 MRLs and MRGs for Semantic Parsing

Figure 1 (a) gives an example of a natural language sentence and its corresponding MR in an MRL called CLANG which is a formal declar-

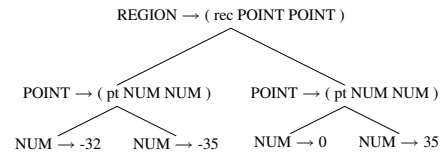


Figure 2: The parse for the CLANG expression “(rec (pt -32 -35) (pt 0 35))” corresponding to the natural language utterance “our midfield” using its original MRG.

ative language with LISP-like prefix notation designed to instruct simulated soccer players in the RoboCup¹ Coach Competition. The MRL and its MRG was designed by the Coach Competition community (Chen et al., 2003) to suit the requirements of their application independent of how well the MRG conforms with the natural language semantics. They were, in fact, not aware that later (Kate et al., 2005) this will be introduced as a test domain for learning semantic parsers. In this original MRG for CLANG, there are several constructs which do not correspond well with their meanings in the natural language. For example, the MR expression of the rectangle (rec (pt -32 -35) (pt 0 35)) from the example MR in Figure 1 (a), whose parse according to the original MRG is shown in Figure 2, corresponds to the NL utterance “our midfield”. In the parse tree, the nodes are the MRG productions and the tokens in upper-case are non-terminals of the MRG while the tokens in lower-case are terminals of the MRG, this convention will be used throughout the paper. As can be seen, the numbers as well as the productions in the parse of the MR expression do not correspond to anything in its natural language utterance. It is also impossible to derive a semantic parse tree of this MR expression over its natural language utterance because there are not enough words in it to cover all the productions present in the MR parse at the lowest level. To alleviate this problem, the provided MRG was manually modified (Kate et al., 2005) to make it correspond better with the natural language by replacing such long MR expressions for soccer regions by shorter expressions like (midfield our)². This new MRG was used in all the previous work which uses the CLANG corpus. In the next sections of the paper, we will present methods to automatically obtain a

¹<http://www.robocup.org>

²The names for the new tokens introduced were chosen for readability and their similarity to the natural language words is inconsequential for learning semantic parsers.

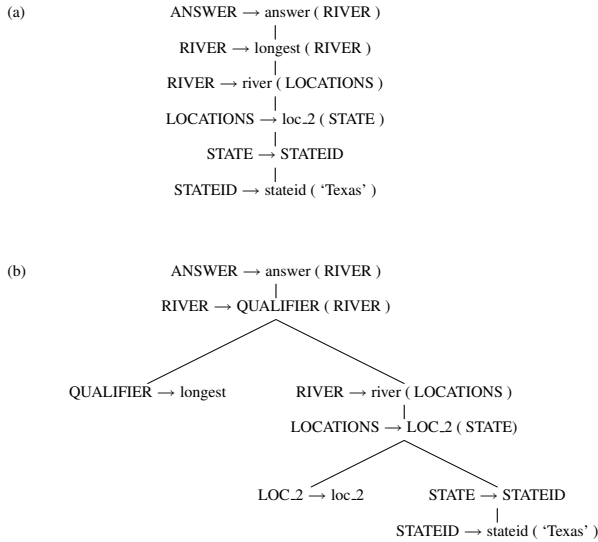


Figure 3: Different parse trees obtained for the MR “answer(longest(river(loc_2(stateid(‘Texas’))))”) corresponding to the NL sentence “Which is the longest river in Texas?” using (a) a simple MRG (b) a manually designed MRG.

better MRG which corresponds well with the NL semantics.

Figure 1 (b) shows an NL sentence and its MR from the GEOQUERY domain (Zelle and Mooney, 1996) which consists of a database of U.S. geographical facts about which a user can query. The MRL used for GEOQUERY in some of the previous work is a variable-free functional query language, that was constructed from the original MRs in Prolog (Kate et al., 2005). From this MRL, the MRG was then manually written so that its productions were compatible with the semantics expressible in natural language. This MRG was different from some simple MRG one would otherwise design for the MRL. Figure 3 (a) shows the parse tree obtained using a simple MRG for the MR shown in Figure 1 (b). The MR parse obtained using the simple MRG is more like a linear chain which means that in a semantic parse of the NL sentence each production will have to correspond to the entire sentence. But ideally, different productions should correspond to the meanings of different substrings of the sentence. Figure 3 (b) shows a parse tree obtained using the manually designed MRG in which the productions QUALIFIER \rightarrow longest and LOC_2 \rightarrow loc_2 would correspond to the semantic concepts of “longest” and “located in” that are expressible in natural language.

Finally, Figure 1 (c) shows the same NL sentence from the GEOQUERY domain but the MR in SQL which is the standard database query lan-

guage. The inner expression finds the length of the longest river in Texas and then the outer expression finds the river in Texas which has that length. Due to space restriction, we are not showing the parse tree for this SQL MR, but its incompatibility with the NL sentence can be seen from the MR itself because part of the query repeats itself with ‘Texas’ appearing twice while in the NL sentence everything is said only once.

2.2 KRISP: A Semantic Parser Learning System

We very briefly describe the semantic parser learning system, KRISP (Kate and Mooney, 2006), which we will use as a base system for transforming MRGs, we however note that the MRG transformation methods presented in this paper are general enough to work with any system which learns semantic parser using MRGs. KRISP (Kernel-based Robust Interpretation for Semantic Parsing) is a supervised learning system for semantic parsing which takes NL sentences paired with their MRs as training data. The productions of the MRG are treated like semantic concepts. For each of these productions, a Support-Vector Machine classifier is trained using string similarity as the kernel (Lodhi et al., 2002). Each classifier can then estimate the probability of any NL substring representing the semantic concept for its production. During semantic parsing, the classifiers are called to estimate probabilities on different substrings of the sentence to compositionally build the most probable MR parse over the entire sentence with its productions covering different substrings of the sentence. KRISP was shown to perform competitively with other existing semantic parser learning systems and was shown to be particularly robust to noisy NL input.

3 Transforming MRGs Using Operators

This section describes an approach to transform an MRG using grammar transformation operators to conform it better with the NL semantics. The following section will present another approach for transforming an MRG using macros which is sometimes more directly applicable.

The MRLs used for semantic parsing are always assumed to be context-free which is true for almost all executable computer languages. There has been some work in learning *context-free grammars* (CFGs) for a language given several exam-

ples of its expressions (Lee, 1996). Most of the approaches directly learn a grammar from the expressions but there also have been approaches that first start with a simple grammar and then transform it using suitable operators to a better grammar (Langley and Stromsten, 2000). The goodness for a grammar is typically measured in terms of its simplicity and coverage. Langley and Stromsten (2000) transform syntactic grammars for NL sentences. To our best knowledge, there is no previous work on transforming MRGs for semantic parsing. For this task, since an initial MRG is always given with the MRL, there is no need to first learn it from its MRs. The next subsection describes the operators our method uses to transform an initial MRG. The subsection following that then describes how and when the operators are applied to transform the MRG during training. Our criteria for goodness of an MRG is the performance of the semantic parser learned using that MRG.

3.1 Transformation Operators

We describe five transformation operators which are used to transform an MRG. Each of these operators preserves the coverage of the grammar, i.e. after application of the operator, the transformed grammar generates the same language that the previous grammar generated³. The MRs do not change but only the way they are parsed may change because of grammar transformations. This is important because the MRs are to be used in an application and hence should not be changed.

1. Create Non-terminal from a Terminal (CreateNT): Given a terminal symbol t in the grammar, this operator adds a new production $T \rightarrow t$ to it and replaces all the occurrences of the terminal t in all the other productions by the new non-terminal T . In the context of semantic parsing learning algorithm, this operator introduces a new semantic concept the previous grammar was not explicit about. For example, this operator may introduce a production (a semantic concept) $\text{LONGEST} \rightarrow \text{longest}$ to the simple grammar whose parse was shown in Figure 3 (a). This is close to the production $\text{QUALIFIER} \rightarrow \text{longest}$ of the manual grammar used in the parse shown in Figure 3 (b).

2. Merge Non-terminals (MergeNT): This operator merges n non-terminals T_1, T_2, \dots, T_n , by introducing n productions $T \rightarrow T_1, T \rightarrow T_2, \dots$,

³This is also known as weak equivalence of grammars.

$T \rightarrow T_n$ where T is a new non-terminal. All the occurrences of the merged non-terminals on the right-hand-side (RHS) of all the remaining productions are then replaced by the non-terminal T . In order to ensure that this operator preserves the coverage of the grammar, before applying it, it is made sure that if one of these non-terminals, say T_1 , occurs on the RHS of a production π_1 then there also exist productions π_2, \dots, π_n which are same as π_1 except that T_2, \dots, T_n respectively occur in them in place of T_1 . If this condition is violated for any production of any of the n non-terminals then this operator is not applicable. This operator enables generalization of some non-terminals which occur in similar contexts leading to generalization of productions in which they occur on the RHS. For example, this operator may generalize non-terminals LONGEST and SHORTEST in GEOQUERY MRG to form $\text{QUALIFIER}^4 \rightarrow \text{LONGEST}$ and $\text{QUALIFIER} \rightarrow \text{SHORTEST}$ productions.

3. Combine Two Non-terminals (CombineNT): This operator combines two non-terminals T_1 and T_2 into one new non-terminal T by introducing a new production $T \rightarrow T_1 T_2$. All the instances of T_1 and T_2 occurring adjacent in this order on the RHS (with at least one more non-terminal⁵) of all the other productions are replaced by the new non-terminal T . For example, the production $A \rightarrow a B T_1 T_2$ will be changed to $A \rightarrow a B T$. This operator will not eliminate other occurrences of T_1 and T_2 on the RHS of other productions in which they do not occur adjacent to each other. In the context of semantic parsing, this operator adds an extra level in the MR parses which does not seem to be useful in itself, but later if the non-terminals T_1 and T_2 get eliminated (by the application of the DeleteProd operator described shortly), this operator will be combining the concepts represented by the two non-terminals.

4. Remove Duplicate Non-terminals (RemoveDuplNT): If a production has the same non-terminal appearing twice on its RHS then this operator adds an additional production which differs from the first production in that it has only one occurrence of that non-terminal. For example, if a production is $A \rightarrow b C D C$, then this operator will introduce a new production $A \rightarrow b C D$ re-

⁴A system generated name will be given to the new non-terminal.

⁵Without the presence of an extra non-terminal on the RHS, this change will merely add redundancy to the parse trees using this production.

moving the second occurrence of the non-terminal C . This operator is applied only when the subtrees under the duplicate non-terminals of the production are often found to be the same in the parse trees of the MRs in the training data. As such this operator will change the MRL the new MRG will generate, but this can be easily reverted by appropriately duplicating the subtrees in its generated MR parses in accordance to the original production. This operator is useful during learning a semantic parser because it eliminates the type of incompatibility between MRs and NL sentences illustrated with Figure 1 (c) in Subsection 2.1.

5. Delete Production (DeleteProd): This last operator deletes a production and replaces the occurrences of its left-hand-side (LHS) non-terminal with its RHS in the RHS of all the other productions. In terms of semantic parsing, this operator eliminates the need to learn a semantic concept. It can undo the transformations obtained by the other operators by deleting the new productions they introduce.

We note that the CombineNT and MergeNT operators are same as the two operators used by Langley and Stromsten (2000) to search a good syntactic grammar for natural language sentences from the space of its possible grammars. We also note that the applications of CreateNT and CombineNT operators can reduce a CFG to its Chomsky normal form⁶, and conversely, because of the reverse transformations achieved by the DeleteProd operator, a Chomsky normal form of a CFG can be converted into any other CFG which accepts the same language.

3.2 Applying Transformation Operators

In order to transform an MRG to improve semantic parsing, since a simple hill-climbing type approach to search the space of all possible MRGs will be computationally very intensive, we use the following error-driven heuristic search which is faster although less thorough.

First, using the provided MRG and the training data, a semantic parser is trained using KRISP. The trained semantic parser is applied to each of the training NL sentences. Next, for each production π in the MRG, two values $total_\pi$ and $incorrect_\pi$ are computed. The value $total_\pi$ counts how many MR parses from the training examples use the produc-

⁶In which all the productions are of the form $A \rightarrow a$ or $A \rightarrow BC$.

tion π . The value $incorrect_\pi$ counts the number of training examples for which the semantic parser incorrectly uses the production π , i.e. it either did not include the production π in the parse of the MR it produces when the correct MR’s parse included it, or it included the production π when it was not present in the correct MR’s parse. These two statistics for a production indicate how well the semantic parser was able to use the production in semantic parsing. If it was not able to use a production π well, then the ratio $incorrect_\pi/total_\pi$, which we call $mistakeRatio_\pi$, will be high indicating that some change needs to be made to that production. After computing these values for all the productions, the procedure described below for applying the first type of operator is followed. After this, the MRs in the training data are re-parsed using the new MRG, the semantic parser is re-trained and the $total_\pi$ and $incorrect_\pi$ values are re-computed. Next, the procedure for applying the next operator is followed and so on. The whole process is repeated for a specified number of iterations. In the experiments, we found that the performance does not improve much after two iterations.

1. Apply CreateNT: For each terminal t in the grammar, $total_t$ and $incorrect_t$ values are computed by summing up the corresponding values for all the productions in which t occurs on the RHS with at least one non-terminal⁷. If $total_t$ is greater than β (a parameter) and $mistakeRatio_t = incorrect_t/total_t$ is greater than α (another parameter), then the CreateNT operator is applied, provided the production $T \rightarrow t$ is not already present.

2. Apply MergeNT: All the non-terminals occurring on the RHS of all those productions π are collected whose $mistakeRatio_\pi$ value is greater than α and whose $total_\pi$ value is greater than β . The set of these non-terminals is then partitioned such that the criteria for applying the MergeNT is satisfied by the non-terminals in each partition with size at least two. The MergeNT operator is then applied to the non-terminals in each partition with size at least two.

3. Apply CombineNT: For every non-terminal pair T_1 and T_2 , $total_{T_1T_2}$ and $incorrect_{T_1T_2}$ values are computed by summing their corresponding values for the productions in which the two non-terminals are adjacent in the RHS in the

⁷Without a non-terminal on the RHS, the operator will only add a redundant level to the parses which use this production.

presence of at least one more non-terminal. If $mistakeRatio_{T_1 T_2} = incorrect_{T_1 T_2} / total_{T_1 T_2}$ is greater than α and $total_{T_1 T_2}$ is greater than β , then the CombineNT operator is applied to these two non-terminals.

4. Apply RemoveDuplNT: If a production π has duplicate non-terminals on the RHS under which the same subtrees are found in the MR parse trees of the training data more than once then this operator is applied provided its $mistakeRatio_\pi$ is greater than α and $total_\pi$ is greater than β .

5. Apply DeleteProd: The DeleteProd operator is applied to all the productions π and whose $mistakeRatio_\pi$ is greater than α and $total_\pi$ is greater than β . This step simply deletes the productions which are mostly incorrectly used.

For the experiments, we set the α parameter to 0.75 and β parameter to 5, these values were determined through pilot experiments.

4 Transforming MRGs Using Macros

As was illustrated with Figure 2 in Subsection 2.1, sometimes there can be large parses for MR expressions which do not correspond well with their semantics in the natural language. While it is possible to transform the MRG using the operators described in the previous section to reduce a subtree of the parse to just one production which will then correspond directly to its meaning in the natural language, it will require a particular sequence of transformation operators to achieve this which may rarely happen during the heuristic search used in the MRG transformation algorithm. In this section, we describe a more direct way of obtaining such transformations using macros.

4.1 Meaning Representation Macros

A meaning representation macro for an MRG is a production formed by combining two or more existing productions of the MRG. For example, for the CLANG example shown in Figure 2, the production $REGION \rightarrow (rec(pt -32 -35)(pt 0 35))$ is a meaning representation macro. There could also be non-terminals on its RHS. From an MR parse drawn with non-terminals at the internal nodes (instead of productions), a macro can be derived from a subtree⁸ rooted at any of the internal nodes by making its root as the LHS non-terminal and the left-to-right sequence formed by its leaves (which could

⁸Each node of a subtree must either include all the children nodes of the corresponding node from the original tree or none of them.

be non-terminals) as the RHS. We use the following error-driven procedure to introduce macros in the MRG in order to improve the performance of semantic parsing.

4.2 Learning Meaning Representation Macros

A semantic parser is first learned from the training data using KRISP and the given MRG. The learned semantic parser is then applied to the training sentences and if the system can not produce any parse for a sentence then the parse tree of its corresponding MR is included in a set called *failed parse trees*. Common subtrees in these failed parse trees are likely to be good candidates for introducing macros. Then a set of *candidate trees* is created as follows. This set is first initialized to the set of failed parse trees. The largest common subtree of every pair of trees in the candidate trees is then also included in this set if it is not an empty tree. The process continues with the newly added trees until no new tree can be included. This process is similar to the repeated bottom-up generalization of clauses used in the inductive logic programming system GOLEM (Muggleton and Feng, 1992). Next, the trees in this set are sorted based on the number of failed parse trees of which they are a subtree. The trees which are part of fewer than β subtrees are removed. Then in highest to lowest order, the trees are selected one-by-one to form macros, provided their height is greater than two (otherwise it will be an already existing MRG production) and an already selected tree is not its subtree. A macro is formed from a tree by making the non-terminal root of the tree as its LHS non-terminal and the left-to-right sequence of the leaves as its RHS.

These newly formed macros (productions) are then included in the MRG. The MRs in the training data are re-parsed and the semantic parser is re-trained using the new MRG. In order to delete the macros which were not found useful, a procedure similar to the application of DeleteProd is used. The $total_\pi$ and $incorrect_\pi$ values for all the macros are computed in a manner similar to described in the previous section. The macros for which $mistakeRatio_\pi = total_\pi / incorrect_\pi$ is greater than α and $total_\pi$ is greater than β are removed. This whole procedure of adding and deleting macros is repeated a specified number of iterations. In the experiments, we found that two

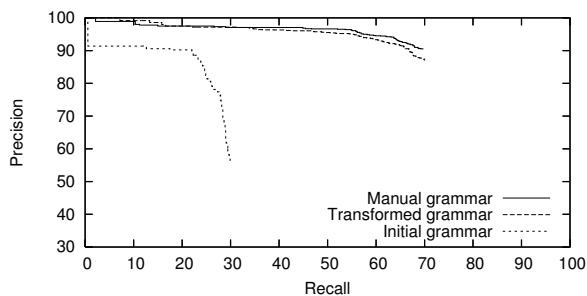


Figure 4: The results comparing the performances of the learned semantic parsers on the GEOQUERY domain with the functional query language using different MRGs.

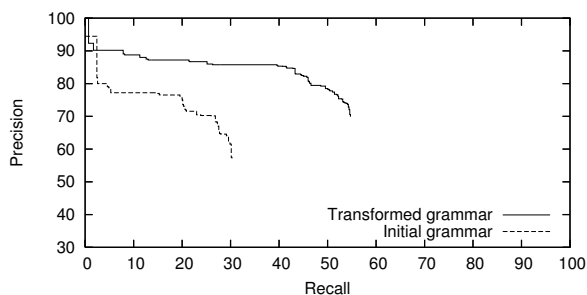


Figure 5: The results comparing the performances of the learned semantic parsers on the GEOQUERY domain with SQL as the MRL using different MRGs.

iterations are usually sufficient.

5 Experiments

We tested our MRG transformation methods with MRGs of three different MRLs which were described in the Background section. In each case, we first transformed the given MRG using macros and then using grammar transformation operators. The training and testing was done using standard 10-fold cross-validation and the performance was measured in terms of precision (the percentage of generated MRs that were correct) and recall (the percentage of all sentences for which correct MRs were obtained). Since we wanted to evaluate how the grammar transformation changes the performance on the semantic parsing task, in each of the experiments, we used the same system, KRISP, and compared how it performs when trained using different MRGs for the same MRL. Since KRISP assigns confidences to the MRs it generates, an entire range of precision-recall trade-off was plotted by measuring precision and recall values at various confidence levels.

Figure 4 shows the results on the GEOQUERY domain using the functional query language whose

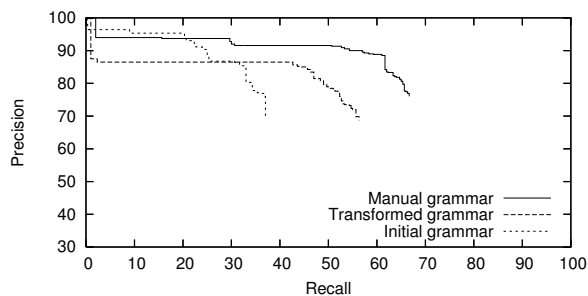


Figure 6: The results comparing the performances of the learned semantic parsers on the CLANG corpus using different MRGs.

corpus contained total 880 NL-MR pairs. As can be seen, the performance of the semantic parser that KRISP learns when trained using the initial simple MRG for the MRL is not good. But when that MRG is transformed, the performance of the semantic parser dramatically improves and is very close to the performance obtained with the manually-engineered grammar. The macro transformations did not help improve the performance with this MRG, and most of the performance gain was obtained because of the CreateNT and DeleteProd operators.

We next tested our MRG transformation algorithm on SQL as the MRL for the GEOQUERY domain. This corpus contains 700 NL-MR pairs in which the NL sentences were taken from the original 880 examples. This corpus was previously used to evaluate the PRECISION system (Popescu et al., 2003), but since that system is not a machine learning system, its results cannot be directly compared with ours. The initial MRG we used contained the basic SQL productions. Figure 5 shows that results improve by a large amount after MRG transformations. We did not have any manually-engineered MRG for SQL for this domain available to us. With this MRG, most of the improvement was obtained using the macros and the RemoveDuplNT transformation operator.

Finally, we tested our MRG transformation algorithm on the CLANG domain using its original MRG in which all the chief regions of the soccer field were in the form of numeric MR expressions which do not correspond to their meanings in the natural language. Its corpus contains 300 examples of NL-MR pairs. Figure 6 shows the results. After applying the MRG transformations the performance improved by a large margin. The gain was due to transformations obtained us-

ing macros while the grammar transformation operators did not help with this MRG. Although the precision was lower for low recall values, the recall increased by a large quantity and the best F-measure improved from 50% to 63%. But the performance still lagged behind that obtained using the manually-engineered MRG. The main reason for this is that the manual MRG introduced some domain specific expressions, like `left`, `right`, `left-quarter` etc., which correspond directly to their meanings in the natural language. On the other hand, the only way to specify “left” of a region using the original CLANG MRG is by specifying the coordinates of the left region, like `(rec (pt -32 -35) (pt 0 0))` is the left of `(rec (pt -32 -35) (pt 0 35))` etc. It is not possible to learn the concept of “left” from such expressions even with MRG transformations.

6 Conclusions

A meaning representation grammar which does not correspond well with the natural language semantics can lead to a poor performance by a learned semantic parser. This paper presented grammar transformation operators and meaning representation macros using which the meaning representation grammar can be transformed to make it better conform with the semantics of natural language. Experimental results on three different grammars demonstrated that the performance on semantic parsing task can be improved by large amounts by transforming the grammars.

Acknowledgments

I would like to thank Raymond Mooney for many useful discussions regarding the work described in this paper.

References

- Chen et al. 2003. Users manual: RoboCup soccer server manual for soccer server version 7.07 and later. Available at <http://sourceforge.net/projects/sserver/>.
- Ge, R. and R. J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proc. of CoNLL-2005*, pages 9–16, Ann Arbor, MI.
- Kate, R. J. and R. J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proc. of COLING/ACL-2006*, pages 913–920, Sydney, Australia.
- Kate, R. J., Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In *Proc. of AAAI-2005*, pages 1062–1068, Pittsburgh, PA.
- Langley, Pat and Sean Stromsten. 2000. Learning context-free grammar with a simplicity bias. In *Proc. of ECML-2000*, pages 220–228, Barcelona, Spain.

- Lee, Lillian. 1996. Learning of context-free languages: A survey of the literature. Technical Report TR-12-96, Center for Research in Computing Technology, Harvard University.
- Lodhi, Huma, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.
- Muggleton, Stephen and C. Feng. 1992. Efficient induction of logic programs. In Muggleton, Stephen, editor, *Inductive Logic Programming*, pages 281–297. Academic Press, New York.
- Nguyen, Le-Minh, Akira Shimazu, and Xuan-Hieu Phan. 2006. Semantic parsing with structured SVM ensemble classification models. In *Proc. of COLING/ACL 2006 Main Conf. Poster Sessions*, pages 619–626, Sydney, Australia.
- Popescu, Ana-Maria, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proc. of IUI-2003*, pages 149–157, Miami, FL.
- Tang, L. R. and R. J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proc. of ECML-2001*, pages 466–477, Freiburg, Germany.
- Wong, Y. W. and R. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proc. of HLT/NAACL-2006*, pages 439–446, New York City, NY.
- Wong, Y. W. and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proc. of ACL-2007*, pages 960–967, Prague, Czech Republic.
- Zelle, J. M. and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proc. of AAAI-1996*, pages 1050–1055, Portland, OR.
- Zettlemoyer, Luke S. and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proc. of EMNLP-CoNLL-2007*, pages 678–687, Prague, Czech Republic.

Using LDA to detect semantically incoherent documents

Hemant Misra and Olivier Cappé
LTCI/CNRS and TELECOM ParisTech
{misra,cappe}@enst.fr

François Yvon
Univ Paris-Sud 11 and LMISI-CNRS
yvon@limsi.fr

Abstract

Detecting the semantic coherence of a document is a challenging task and has several applications such as in text segmentation and categorization. This paper is an attempt to distinguish between a ‘semantically coherent’ true document and a ‘randomly generated’ false document through topic detection in the framework of latent Dirichlet analysis. Based on the premise that a true document contains only a few topics and a false document is made up of many topics, it is asserted that the entropy of the topic distribution will be lower for a true document than that for a false document. This hypothesis is tested on several false document sets generated by various methods and is found to be useful for fake content detection applications.

1 Introduction

The “Internet revolution” has dramatically increased the monetary value of higher ranking on the web search engines index, fostering the expansion of techniques, collectively known as “Web Spam”, that fraudulently help to do so. Internet is indeed “polluted” with fake Web sites whose only purpose is to deceive the search engines by artificially pushing up the popularity of commercial sites, or sites promoting illegal content¹. These fake sites are often forged using very crude content generation techniques, ranging from *web scraping* (blending of chunks of actual contents) to simple-minded text generation techniques based

on random sampling of words (“word salads”), or randomly replacing words in actual documents (“word stuffing”)². Among these, the latter two are easy to detect using simple statistical models of natural texts, but the former is more challenging, it being made up of actual sentences: recognizing these texts as forged requires either to resort to plagiarism detection techniques, or to automatically identify their lack of *semantic consistency*.

Detecting the consistency of texts or of text chunks has many applications in Natural Language Processing. So far, it has been used mainly in the context of automatic text segmentation, where a change in vocabulary is often the mark of topic change (Hearst, 1997), and, to a lesser extent, in discourse studies (see, e.g., (Foltz et al., 1998)). It could also serve to devise automatic metrics for text summarization or machine translation tasks.

This paper is an attempt to address the issue of differentiating between ‘true’ and ‘false’ documents on the basis of their consistency through topic modeling approach. We have used Latent Dirichlet allocation (LDA) (Blei et al., 2002) model as our main topic modeling tool. One of the aims of LDA and similar methods, including probabilistic latent semantic analysis (PLSA) (Hofmann, 2001), is to produce low dimensionality representations of texts in a “semantic space” such that most of their inherent statistical characteristics are preserved. A reduction in dimensionality facilitates storage as well as faster retrieval. Modeling discrete data has many applications in classification, categorization, topic detection, data mining, information retrieval (IR), summarization and collaborative filtering (Buntine and Jakulin, 2004).

The aim of this paper is to test LDA for establishing the semantic coherence of a document based on the premise that a real (coherent) document should discuss only a few number of topics,

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

¹The annual AirWeb challenge <http://airweb.cse.lehigh.edu> gives a state-of-the art on current Web Spam detection techniques.

²The same techniques are commonly used in mail spams also.

a property hardly granted for forged documents which are often made up of random assemblage of words or sentences. As a consequence, the coherence of a document may reflect in the entropy of its posterior topic distribution or in its perplexity for the model. The entropy of the estimated topic distribution of a true document is expected to be lower than that of a fake document. Moreover, the length normalized log-likelihood of a true and coherent document may be higher as compared to that of a false and incoherent document.

In this paper, we compare two methods to estimate the posterior topic distribution of test documents, and this study is also an attempt to investigate the role of different parameters on the efficiency of these methods.

This paper is organized as follows: In Section 2, the basics of the LDA model are set. We then discuss and contrast several approaches to the problem of inferring the topic distribution of a new document in Section 3. In Section 4, we describe the corpus and experimental set-up that are used to produce the results presented in Section 5. We summarize our main findings and draw perspectives for future research in Section 6.

2 Latent Dirichlet Allocation

2.1 Basics

LDA is a probabilistic model of text data which provides a generative analog of PLSA (Blei et al., 2002), and is primarily meant to reveal hidden topics in text documents. In (Griffiths and Steyvers, 2004), the authors used LDA for identifying “hot topics” by analyzing the temporal dynamics of topics over a period of time. More recently LDA has also been used for unsupervised language model (LM) adaptation in the context of automatic speech recognition (ASR) (Hsu and Glass, 2006; Tam and Schultz, 2007; Heidel et al., 2007). Several extensions of the LDA model, such as hierarchical LDA (Blei et al., 2004), HMM-LDA (Griffiths et al., 2005), correlated topic models (Blei and Lafferty, 2005) and hidden topic Markov models (Gruber et al., 2007), have been proposed, that introduce more complex dependency patterns in the model.

Like most of the text mining techniques, LDA assumes that documents are made up of words and the ordering of the words within a document is unimportant (“bag-of-words” assumption). Contrary to the simpler Multinomial Mixture Model

(see, e.g., (Nigam et al., 2000) and Section 2.4), LDA assumes that every document is represented by a topic distribution and that each topic defines an underlying distribution on words.

The generative history of a document (a bag-of-words) collection is the following: Assuming a fixed and known number of topics n_T , for each topic t , a distribution β_t over the indexing vocabulary ($w = 1 \dots n_W$) is drawn from a Dirichlet distribution. Then, for each document d , a distribution θ_d over the topics ($t = 1 \dots n_T$) is drawn from a Dirichlet distribution. For a document d , the document length l_d being an exogenous variable, the next step consists of drawing a topic t_i from θ_d for each position $i = 1 \dots l_d$. Finally, a word is selected from the chosen topic t_i . Given the topic distribution, each word is thus drawn independently from every other word using a *document specific* mixture model. The probability of i^{th} word token is thus:

$$\begin{aligned} P(w_i|\theta_d, \beta) &= \sum_{t=1}^{n_T} P(t_i=t|\theta_d)P(w_i|t_i, \beta) \quad (1) \\ &= \sum_{t=1}^{n_T} \theta_{dt}\beta_{tw} \quad (2) \end{aligned}$$

Conditioned on β and θ_d , the likelihood of document d is a mere product of terms such as (2), which can be rewritten as:

$$P(C_d|\theta_d, \beta) = \prod_{w=1}^{n_W} \left[\sum_{t=1}^{n_T} (\theta_{dt}\beta_{tw}) \right]^{C_{dw}} \quad (3)$$

where C_{dw} is the count of word w in d .

2.2 LDA: Training

LDA training consists of estimating the following two parameter vectors from a text collection: the topic distribution in each document d ($\theta_{dt}, t = 1 \dots n_T, d = 1 \dots n_D$) and the word distribution in each topic ($\beta_{tw}, t = 1 \dots n_T, w = 1 \dots n_W$). Both θ_d and β_t define discrete distributions, respectively over the set of topics and over the set of words. Various methods have been proposed to estimate LDA parameters, such as variational method (Blei et al., 2002), expectation propagation (Minka and Lafferty, 2002) and Gibbs sampling (Griffiths and Steyvers, 2004). In this paper, we have used the latter approach, which boils down to repeatedly going through the training data and sampling the topic assigned to each word token conditioned

on the topic assigned to all the other word tokens. Given a particular Gibbs sample, the posteriors for θ and β are ³: Dirichlet with parameters $(K_{t1} + \lambda, \dots, K_{tn_W} + \lambda)$ and Dirichlet with parameters $(J_{d1} + \alpha, \dots, J_{dn_T} + \alpha)$, respectively, where K_{tw} is the number of times word w is assigned to topic t and J_{dt} is the number of times topic t is assigned to some word token in document d . Hence,

$$\beta_{tw} = \frac{K_{tw} + \lambda}{\sum_{k=1}^{n_W} K_{tk} + n_W \lambda} \quad (4)$$

$$\theta_{dt} = \frac{J_{dt} + \alpha}{\sum_{k=1}^{n_T} J_{dk} + n_T \alpha} \quad (5)$$

During the Gibbs sampling phase, β_t and θ_d are sampled from the above posteriors while the final estimates for these parameters are obtained by averaging the posterior means over the complete set of Gibbs iteration.

2.3 LDA: Testing

Training LDA model on a text collection already provides interesting insights regarding the thematic structure of the collection. This has been the primary application of LDA in (Blei et al., 2002; Griffiths and Steyvers, 2004). Even better, being a generative model, LDA can be used to make prediction regarding novel documents (assuming they use the same vocabulary as the training corpus). In a typical IR setting, where the main focus is on computing the similarity between a document d and a query d' , a natural similarity measure is given by $P(C_{d'}|\theta_d, \beta)$, computed according to (3) (Buntine et al., 2004).

An alternative would be to compute the KL divergence between the topic distribution in d and d' , which however requires to infer the latter quantity. As the topic distribution of a (new) document gives its representation along the latent semantic dimensions, computing this value is helpful for many applications, including text segmentation and text classification. Methods for efficiently and accurately estimating topic distribution for text documents are presented and evaluated in Section 3.

2.4 Baseline: Multinomial Mixture Model

The performance of LDA model is compared with that of the simpler multinomial mixture model (Nigam et al., 2000; Rigouste et al., 2007).

³assuming non-informative priors with hyper-parameters α and λ for the Dirichlet distribution over topics and the Dirichlet distribution over words respectively

In this model, every word in a document belongs to the same topic, as if the document specific topic distribution θ_d in LDA were bound to lie on one vertex of the $[0, 1]^{n_T}$ simplex. Using the same notations as before (except for θ_t , which now denotes the position independent probability of topic t in the collection), the probability of a document is:

$$P(C_d|\theta_t, \beta) = \sum_{t=1}^{n_T} \theta_t \prod_{w=1}^{n_W} \beta_{tw}^{C_{dw}} \quad (6)$$

This model can be trained through expectation maximization (EM), using the following reestimation formulas, where (7) defines the E-step; (8) and (9) define the M-step.

$$P(t|C_d, \theta, \beta) = \frac{\theta_t \prod_{w=1}^{n_W} (\beta'_{tw})^{C_{dw}}}{\sum_{t=1}^{n_T} \theta_t \prod_{w=1}^{n_W} (\beta_{tw})^{C_{dw}}} \quad (7)$$

$$\theta'_t \propto \alpha + \sum_{d=1}^{n_D} P(t|C_d, \theta, \beta) \quad (8)$$

$$\beta'_{tw} \propto \lambda + \sum_{d=1}^{n_D} C_{dw} P(t|C_d, \theta, \beta) \quad (9)$$

As suggested in (Rigouste et al., 2007), we initialize the EM algorithm by drawing initial topic distributions from a prior Dirichlet distribution with hyper-parameter $\alpha = 1$. $\beta = 0.1$ in all the experiments.

During testing, the parameters of the multinomial models are used to estimate the posterior topic distribution in each document using (7). The likelihood of a test document is given by (6).

3 Inferring the Topic Distribution of Test Documents

$P(C_d|\theta_d)$, the conditional probability of a document d given θ_d is obtained using (3) ⁴. Computing the likelihood of a test document requires to integrate this quantity over θ ; likewise for the computation of the posterior distribution of θ . This integral has no close form solution, but can be approximated using Monte-Carlo sampling techniques as:

$$P(C_d) \approx \frac{1}{M} \sum_{m=1}^M P(C_d|\theta^{(m)}) \quad (10)$$

where $\theta^{(m)}$ denotes the m^{th} sample from the Dirichlet prior, and M is the number of Monte

⁴The dependence on β is dropped for simplicity. β is learned during training and kept fixed during testing.

Carlo samples. Given the typical length of documents and the large vocabulary size, small scale experiments convinced us that a cruder approximation was in order, as the sum in (10) is dominated by the maximum value. We thus contend ourselves to solve:

$$\theta^* = \operatorname{argmax}_{\theta, \sum_t \theta_t = 1} P(C_d | \theta) \quad (11)$$

and use this value to approximate $P(C_d)$ using (3).

The maximization program (11) has no close form solution. However, the objective function is differentiable and log-concave, and can be optimized in a number of ways. We considered two different algorithms: an EM-like approach, initially introduced in (Heidel et al., 2007), and an exponentiated gradient approach (Kivinen and Warmuth, 1997; Globerson et al., 2007).

The first approach implements an iterative procedure based on the following update rule:

$$\theta_{dt} \leftarrow \frac{1}{l_d} \sum_{w=1}^{n_W} \frac{C_{dw} \theta_{dt} \beta_{tw}}{\sum_{t'=1}^{n_T} \theta_{dt'} \beta_{t'w}} \quad (12)$$

Although no justification was given in (Heidel et al., 2007), it can be shown that this update rule converges towards a global optimum of the likelihood. Let θ and θ' be two topic distributions in the n_T -dimensional simplex, $L(\theta) = \log P(C_d | \theta)$, and $\rho_t(w, \theta) = \frac{\theta_t \beta_{tw}}{\sum_{t'} \theta_{t'} \beta_{t'w}}$. We define an auxiliary function $Q(\theta, \theta') = \sum_w C_w (\sum_t \rho_t(w, \theta) \log(\theta'_t))$. $Q(\theta, \theta')$ is concave in θ' , and performs the role played by the auxiliary function in the EM algorithm. Simple calculus suffices to prove that (i) the update (12) maximizes in θ' the function $Q(\theta, \theta')$, and (ii) $Q(\theta, \theta') - Q(\theta, \theta) \geq L(\theta') - L(\theta)$, which stems from the concavity of the log. At an optimum of $Q(\theta, \theta')$ the positivity of the first term implies the positivity of the second. Maximizing Q using the update rule (12) thus increases the likelihood and repeating this update converges towards the optimum value. We experimented both with an unsmoothed (12) and with a smoothed version of this update rule. The unsmoothed version yielded a slightly better result than the smoothed one.

Exponentiated gradient (Kivinen and Warmuth, 1997; Globerson et al., 2007) yields an alternative update rule:

$$\theta_{dt} \leftarrow \theta_{dt} \exp \left(\eta \sum_{w=1}^{n_W} \frac{C_{dw} \beta_{tw}}{\sum_{t'=1}^{n_T} \theta_{dt'} \beta_{t'w}} \right) \quad (13)$$

where η defines the convergence rate. In this form, the update rule does not preserve the normalization of θ , which needs to be performed after every iteration.

A systematic comparison of these rules was carried out, yielding the following conclusions:

- the convergence of the EM-like method is very fast. Typically, it requires less than half a dozen iterations to converge. After convergence, the topic distribution estimated by this method for a subset of train documents was always very close (as measured by the KL-divergence) to the respective topic distribution of the same documents observed at the end of the LDA training. Taking $n_T = 50$, the average KL divergence for a set of 4,500 documents was found to be less than 0.5.
- exponentiated gradient has a more erratic behaviour, and requires a careful tuning of η on a *per document basis*. For large values of η , the update rule (13) sometimes fails to converge; smaller values of η allowed to consistently reach convergence, but required more iterations (typically 20-30). On a positive side, on an average, the topic distributions estimated by this method are better than the ones obtained with the EM-like algorithm.

Based on these findings, we decided to use the EM-like algorithm in all our subsequent experiments.

4 Experimental protocol

4.1 Training and test corpora

The Reuters Corpus Volume 1 (RCV1) (Lewis et al., 2004) is a collection of over 800,000 news items in English from August 1996 to August 1997. Out of the entire RCV1 dataset, we selected 27,672 documents (news items) for training (**TrainReuters**) and 23,326 documents for testing (**TestReuters**). The first 4000 documents from the **TestReuters** dataset were used as true documents (**TrueReuters**) in the experiments reported in this paper. The vocabulary size in the train set, after removing the function words, is 93,214.

Along with these datasets of “true” documents, three datasets of fake documents were also created. Document generation techniques are many: here we consider documents made by mixing short passages from various texts and documents made

by assembling randomly chosen words (sometimes called as “word salads”). In addition, we also consider the case of documents generated with a stochastic language model (LM). Our “fake” test documents are thus composed of:

- (**SentenceSalad**) obtained by randomly picking sentences from **TestReuters**.
- (**WordSalad**) created by generating random sentences from a conventional unigram LM trained on **TrainReuters**.
- (**Markovian**) created by generating random sentences from a conventional 3-gram LM trained on **TrainReuters**.

Each of these forged document set contains 4,000 documents.

To assess the performance on out-of-domain data, we replicated the same tests using 2,000 Medline abstracts (Ohta et al., 2002). 1,500 documents were used either to generate fake documents by picking sentences randomly or to train an LM and then using the LM to generate fake documents. The remaining 500 abstracts were set aside as “true” documents (**TrueMedline**).

4.2 Performance Measurements : EER

The entropy of the topic distribution is computed as $H = -\sum_{j=1}^T \hat{\theta}_{aj} \log \hat{\theta}_{aj}$. The other measure of interest is the average ‘log-likelihood per word’ (LLPW)⁵.

While evaluating the performance of our system, two types of errors are encountered: false acceptance (FA) when a false document is accepted as a true document and false rejection (FR) when a true document is rejected as a false document. The rate of FA and FR is dependent on the threshold used for taking the decision, and usually the performance of a system is shown by its receiver operating characteristic (ROC) curve which is a plot between FA and FR rates for different values of threshold. Instead of reporting the performance of a system based on two error rates (FA and FR), the general practice is to report the performance in terms of equal-error-rate (EER). The EER is the error rate at the threshold where FA rate = FR rate.

In our system, a threshold on entropy (or LLPW) is used for taking the decision, and all the

⁵This measure is directly related to the text perplexity in the model, according to perplexity = $2^{-\text{average log-likelihood per word}}$

documents having their entropy (or LLPW) below (or above) the threshold are accepted as true documents. The EER is obtained on the test set by changing the threshold on the test set itself, and the best results thus obtained are reported.

5 Detecting semantic inconsistency

5.1 Detecting fake documents with LDA and Multinomial mixtures

In the first set of experiments, the LLPW and entropy of the topic distribution (the two measures) of the Multinomial mixture and LDA models were compared to check the ability of these two measures and models in discriminating between true and false documents. These results are summarized in Table 1.

TrueReuters vs.	Multinomial	
	LLPW	Entropy
SentenceSalad	15.3%	48.8%
WordSalad	9.3%	35.8%
Markovian	17.6%	38.9%
TrueReuters vs.	LDA	
	LLPW	Entropy
SentenceSalad	18.9%	0.88%
WordSalad	9.9%	0.13%
Markovian	25.0%	0.28%

Table 1: *Performance of the Multinomial Mixture and LDA*

For the multinomial mixture model, the LLPW measure is able to discriminate between true and false documents to a certain extent. As expected (not shown here), the LLPW of the true documents is usually higher than that of the false documents. In contrast, the entropy of the posterior topic distribution does not help much in discriminating between true and false documents. In fact it remains close to zero (meaning that only one topic is “active”) both for true and false documents.

The behaviour of the LDA scores is entirely different. The perplexity scores (LLPW) of true and fake texts are comparable, and do not make useful predictors. In contrast, the entropy of the topic distribution allows to sort true documents from fake ones with a very high accuracy for all kinds of fake texts considered in this paper. Both results stem from the ability of LDA to assign a different topic to each word occurrence.

Similar pattern is observed for our three false test sets (against the TrueReuters set) with small

variations The texts generated with a Markov model, no matter the order, have the highest entropy, reflecting the absence of long range correlation in the generation model. Though the texts generated by mixing sentences are more confusing with the true documents, the performance is still less than 1% EER. Texts mixing a high number of topics (e.g., Sentence Salads) are almost as likely as natural texts that address only a few topics. However, the former has much higher entropy of the topic distribution due to a large number of topics being active in such texts (see also Figure 1).

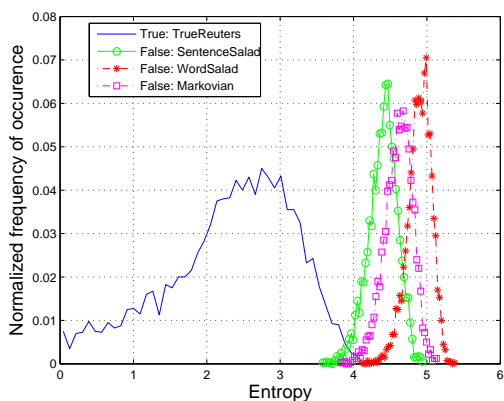


Figure 1: Histogram of entropy of θ for different true and false document sets.

It is noteworthy that both the predictors (LLPW and Entropy) give complementary clues regarding a text category. A linear combination of these two scores (the weight to the LLPW score is 0.1) allows to substantially improve over these baseline results, yielding a relative improvement (in EER) of +20.0% for the sentence salads, +20.8% for the word salads, and +27.3% for the Markov Models.

5.2 Effect of the number of topics

In this part, we investigate the performance of LDA in detecting false documents when the number of topics is changed. Increasing the number of topics means higher memory requirements both during training and testing. Though the results are shown only for SentenceSalad, similar trend is observed for WordSalad and Markovian.

The numbers in Table 2 show that the performance obtained with the LLPW score consistently improve with an increase in the number of topics, though the % improvement obtained when the

number of topics exceeds 200 is marginal. In contrast, the best performance in case of entropy is achieved at 50 topics and slowly degrades when a more complex model is used.

Number of Topics	LLPW	Entropy
10	27.9	1.88
50	18.9	0.88
100	16.0	0.93
200	14.8	0.90
300	13.8	1.05
400	13.6	1.10

Table 2: EER from LLPW and Entropy distribution for TrueReuters against SentenceSalad.

5.3 Detecting “noisy” documents

In this section, we study fake documents produced by randomly changing words in true documents (the TrueReuters dataset). In each document, a fixed percentage of content words is randomly replaced by any other word from the training vocabulary⁶. This percentage was varied from 5 to 100 and EER for these corrupted document sets is computed at each % corruption level (Figure 2). As

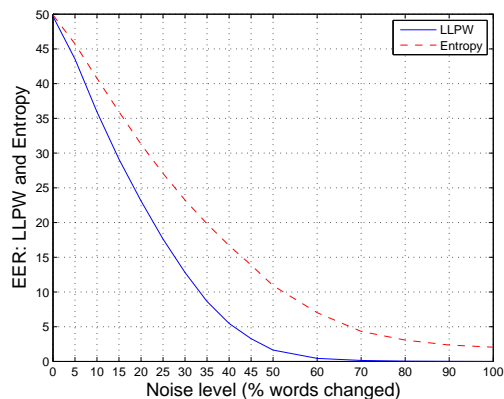


Figure 2: EER at various noise levels

expected, the EER is very high at low noise levels, and as the noise level is increased, EER gets lower. When only a few words are changed in a true document, it retains the properties of a true document (high LLPW and low entropy). However, as more number of words are changed in a true document,

⁶When the replacement words are chosen from a small set of very specific words, the fake document generation strategy is termed as “word stuffing”.

it starts showing the characteristics of a false document (low LLPW and high entropy). These results suggest that our semantic consistency tests are too crude a measure to detect a small number of inconsistencies, such as the ones found in the state-of-the-art OCR or ASR systems’ outputs. On the other hand, it confirms the numerous studies that have shown that topic detection (and topic adaptation) or text categorization tasks can be performed with the same accuracy for moderately noisy texts and clean texts, a finding which warrants the topic-based LM adaptation strategies deployed in (Heidel et al., 2007; Tam and Schultz, 2007).

The difference in the behavior of our two predictors is striking. The EER obtained using LLPW drops more quickly than the one obtained with entropy of the topic distribution. It suggests that the influence of “corrupting” content words (mostly with low β_{tw}) is heavy on the LLPW, but the topic information is not lost till a majority of the “uncorrupted” content words belong to the same topic.

5.4 Effect of the document length

In this section, we study the robustness of our two predictors with respect to the document length by progressively increasing the number of content words in a document (true or fake). As can be seen from Figure 3, the entropy of the posterior topic distribution starts to provide a reasonable discrimination (5% EER) when the test documents contain about 80 to 100 content words, and attains results comparable to those reported earlier in this paper when this number doubles. This definitely rules out this method as a predictor of the semantic consistency of a sentence: we need to consider at least a paragraph to get acceptable results.

5.5 Testing with out-of-domain data

In this section, we study the robustness of our predictors on out-of-domain data using a small excerpt of abstracts from the Medline database. Both true and fake documents are from this dataset. The results are summarized in Table 3. The per-

TrueMedline vs.	LLPW	Entropy
SentenceSalad	31.23%	22.13%
WordSalad	30.03%	19.46%
Markovian	36.51%	23.63%

Table 3: Performance of LDA on PubMed abstracts

formance on out-of-domain documents is poor,

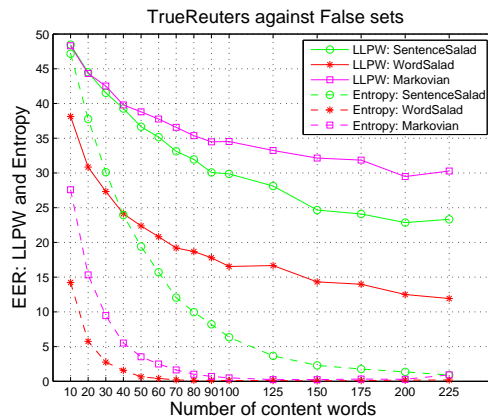


Figure 3: EER with change in number of content words used for LDA analysis. EER based on: LLPW of TrueReuters and false document sets (solid line) and Entropy of topic distribution of TrueReuters and false document sets (dashed line).

though the entropy of the topic distribution is still the best predictor. The reasons for this failure are obvious: a majority of the words occurring in these documents (true or fake) are, from the perspective of the model, characteristic of one single Reuters topic (health and medicine). They cannot be distinguished either in terms of perplexity or in terms of topic distribution (the entropy is low for all the documents). It is interesting to note that all the out-of-domain Medline data can be separated from the in-domain TrueReuters data with good accuracy on the basis of the lower LLPW of the former as compared to the higher LLPW of the latter.

6 Conclusion

In the LDA framework, this paper investigated two methods to infer the topic distribution in a test document. Further, the paper suggested that the coherence of a document can be evaluated based on its topic distribution and average LLPW, and these measures can help to discriminate between true and false documents. Indeed, through experimental results, it was shown that entropy of the topic distribution is lower and average LLPW of true documents is higher for true documents and the former measure was found to be more effective. However, the poor performance of this method on out-of-domain data suggests that we need to use a much larger training corpus to build a robust fake document detector. This raises the issue of train-

ing LDA model with very large collections. In future we would like to explore the potential of this method for text segmentation tasks.

Acknowledgment

This research was supported by the European Commission under the contract *FP6-027026-K-Space*. The views expressed in this paper are those of the authors and do not necessarily represent the views of the commission.

References

- Blei, David and John Lafferty. 2005. Correlated topic models. In *Advances in Neural Information Processing Systems (NIPS'18)*, Vancouver, Canada.
- Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2002. Latent Dirichlet allocation. In Dietterich, Thomas G., Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 601–608, Cambridge, MA. MIT Press.
- Blei, David M., Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. 2004. Hierarchical topic models and the nested Chinese restaurant process. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, Vancouver, Canada.
- Buntine, Wray and Aleks Jakulin. 2004. Applying discrete PCA in data analysis. In Chickering, M. and J. Halpern, editors, *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI'04)*, pages 59–66. AUAI Press 2004.
- Buntine, Wray, Jaakko Löfström, Jukka Perkiö, Sami Perttu, Vladimir Poroshin, Tomi Silander, Henry Tirri, Antti Tuominen, and Ville Tuulos. 2004. A scalable topic-based open source search engine. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 228–234, Beijing, China.
- Foltz, P.W., W. Kintsch, and T.K. Landauer. 1998. The measurement of textual coherence with Latent Semantic Analysis. *Discourse Processes*, 25(2-3):285–307.
- Globerson, Amir, Terry Y. Koo, Xavier Carreras, and Michael Collins. 2007. Exponentiated gradient algorithms for log-linear structured prediction. In *International Conference on Machine Learning*, Corvallis, Oregon.
- Griffiths, Thomas L. and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101 (supl 1):5228–5235.
- Griffiths, Thomas L., Mark Steyvers, David M. Blei, and Joshua Tenenbaum. 2005. Integrating topics and syntax. In *Proceedings of NIPS, 17*, Vancouver, CA.
- Gruber, Amit, Michal Rosen-Zvi, and Yair Weiss. 2007. Hidden topic Markov models. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, San Juan, Puerto Rico, March.
- Hearst, Marti. 1997. TextTiling: Segmenting texts into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.
- Heidel, Aaron, Hung an Chang, and Lin shan Lee. 2007. Language model adaptation using latent Dirichlet allocation and an efficient topic inference algorithm. In *Proceedings of European Conference on Speech Communication and Technology*, Antwerp, Belgium.
- Hofmann, Thomas. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning Journal*, 42(1):177–196.
- Hsu, Bo-June (Paul) and Jim Glass. 2006. Style & topic language model adaptation using HMM-LDA. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia.
- Kivinen, Jyrki and Manfred K. Warmuth. 1997. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132:1–63.
- Lewis, David D., Yiming Yang, Tony Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Machine Learning Research*, 5:361–397.
- Minka, Thomas and John Lafferty. 2002. Expectation-propagation for the generative aspect model. In *Proceedings of the conference on Uncertainty in Artificial Intelligence (UAI)*.
- Nigam, K., A. K. McCallum, S. Thrun, and T. M. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134.
- Ohta, Tomoko, Yuka Tateisi, Hideki Mima, Jun ichi Tsujii, and Jin-Dong Kim. 2002. The GENIA corpus: an annotated research abstract corpus in molecular biology domain. In *Proceedings of Human Language Technology Conference*, pages 73–77.
- Rigouste, Loïs, Olivier Cappé, and François Yvon. 2007. Inference and evaluation of the multinomial mixture model for text clustering. *Information Processing and Management*, 43(5):1260–1280, September.
- Tam, Yik-Cheung and Tanja Schultz. 2007. Correlated latent semantic model for unsupervised LM adaptation. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Honolulu, Hawaii, U.S.A.

Picking them up and Figuring them out: Verb-Particle Constructions, Noise and Idiomaticity

Carlos Ramisch^{♣◇}, Aline Villavicencio^{♣♠}, Leonardo Moura[♣] and Marco Idiart[♡]

[♣]Institute of Informatics, Federal University of Rio Grande do Sul (Brazil)

[◇]GETALP Laboratory, Joseph Fourier University - Grenoble INP (France)

[♠]Department of Computer Sciences, Bath University (UK)

[♡]Institute of Physics, Federal University of Rio Grande do Sul (Brazil)

{ceramisch, avillavicencio, lfsmoura}@inf.ufrgs.br, idiart@if.ufrgs.br

Abstract

This paper investigates, in a first stage, some methods for the automatic acquisition of verb-particle constructions (VPCs) taking into account their statistical properties and some regular patterns found in productive combinations of verbs and particles. Given the limited coverage provided by lexical resources, such as dictionaries, and the constantly growing number of VPCs, possible ways of automatically identifying them are crucial for any NLP task that requires some degree of semantic interpretation. In a second stage we also study whether the combination of statistical and linguistic properties can provide some indication of the degree of idiomaticity of a given VPC. The results obtained show that such combination can successfully be used to detect VPCs and distinguish idiomatic from compositional cases.

1 Introduction

Considerable investigative effort has focused on the automatic identification of Multiword Expressions (MWEs), like compound nouns (*science fiction*) and phrasal verbs (*carry out*) (e.g. Pearce (2002), Evert and Krenn (2005) and Zhang et al. (2006)). Some of them employ language and/or type dependent linguistic knowledge for the task, while others employ independent statistical methods, such as Mutual Information and Log-likelihood (e.g. Pearce (2002) and, Zhang et al. (2006)), or even a combination of them (e.g.

Baldwin (2005) and Sharoff (2004)), as basis for helping to determine whether a given sequence of words is in fact an MWE. Although some research aims at developing methods for dealing with MWEs in general (e.g. Zhang et al. (2006), Ramisch et al. (2008)), there is also some work that deals with specific types of MWEs (e.g. Pearce (2002) on collocations and Villavicencio (2005) on verb-particle constructions (VPCs)) as each of these MWE types has distinct distributional and linguistic characteristics.

VPCs are combinations of verbs and particles, such as *take off* in *Our plane took off late*, that due to their complex characteristics and flexible nature, provide a real challenge for NLP. In particular, there is a lack of adequate resources to identify and treat them, and those that are available provide only limited coverage, in face of the huge number of combinations in use. For tasks like parsing and generation, it is essential to know whether a given VPC is possible or not, to avoid for example using combinations that sound unnatural or ungrammatical to native speakers (e.g. *give/lend/?grant out for the conveying of something to someone or some place* - (Fraser, 1976)).¹ Thus, the knowledge of which combinations are possible is crucial for precision grammar engineering. In addition, as the semantics of VPCs varies from the idiomatic to the more compositional cases, methods for the automatic detection and handling of idiomaticity are very important for any NLP task that involves some degree of semantic interpretation such as Machine Translation (in this case avoiding the problem of producing an unrelated translation for a source sentence). Automatic methods for the identification of idiomaticity in MWEs have been

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

¹See Baldwin et al. (2004) for a discussion of the effects of multiword expressions like VPCs on a parser's performance.

proposed using a variety of approaches such as statistical, substitutional, distributional, etc. (e.g. McCarthy et al. (2003), Bannard (2005) and Fazly and Stevenson (2006)). In particular, Fazly and Stevenson (2006) look at the correlation between syntactic fixedness (in terms of e.g. passivisation, choice of determiner type and pluralisation) and non-compositionality of verb-noun compounds such as *shoot the breeze*.

In this work we investigate the automatic extraction of VPCs, looking into a variety of methods, combining linguistic with statistical information, ranging from frequencies to association measures: Mutual Information (MI), χ^2 and Entropy. We also investigate the determination of compositionality of VPCs verifying whether the degree of semantic flexibility of a VPC combined with some statistical information can be used to determine if it is idiomatic or compositional.

This paper starts with a brief description of VPCs, research on their automatic identification and determination of their semantics (§ 2). We then explain the research questions and the assumptions that serve as the basis for the application of statistical measures (§ 3) on the dataset (§ 4). Our methods and experiments are then detailed (§ 5), and the results obtained are analysed (§ 6). We conclude with a discussion of the contributions that this work brings to the research on verb-particle constructions (§ 7).

2 Verb-Particle Constructions in Theory and Practice

Particles in VPCs are characterised by containing features of motion-through-location and of completion or result in their core meaning (Bolinger, 1971). VPCs can range from idiosyncratic or semi-idiosyncratic combinations, such as *get on* (in e.g. *Bill got on well with his new colleagues*), to more regular ones, such as *tear up* (e.g. in *In a rage she tore up the letter Jack gave her*). A three way classification is adopted by (Dehé, 2002) and (Jackendoff, 2002), where a VPC can be classified as compositional, idiomatic or aspectual, depending on its sense. In compositional VPCs the meaning of the construction is determined by the literal interpretations of the particle and the verb. These VPCs usually involve particles with directional or spatial meaning, and these can often be replaced by the appropriate directional PPs (e.g. *carry in* in *Sheila carried the bags in/into the house* Dehé

(2002)). Idiomatic VPCs, on the other hand, cannot have their meaning determined by interpreting their components literally (e.g. *get on*, meaning *to be on friendly terms with someone*). The third class is that of aspectual VPCs, which have the particle providing the verb with an endpoint, suggesting that the action described by the verb is performed completely, thoroughly or continuously (e.g. *tear up* meaning *to tear something into a lot of small pieces*).

From a syntactic point of view, a given combination can occur in several different subcategorisation frames. For example, *give up* can occur as an intransitive VPC (e.g. in *I give up! Tell me the answer*), where no other complement is required, or it may occur as a transitive VPC which requires a further NP complement (e.g. in *She gave up alcohol while she was pregnant*). Since in English particles tend to be homographs with prepositions (*up, out, in*), a verb followed by a preposition/particle and an NP can be ambiguous between a transitive VPC and a prepositional verb (e.g. *rely on*, in *He relies on his wife for everything*). Some criteria that characterise VPCs are discussed by Bolinger (1971):²

- C1** In a transitive VPC the particle may come either before or after the NP (e.g. *He backed up the team* vs. *He backed the team up*). However, whether a particle can be separated or not from the verb may depend on the degree of bonding between them, the size of the NP, and the kind of NP. This is considered by many to be sufficient condition for diagnosing a VPC, as prepositions can only appear in a position contiguous to the verb (e.g. **He got the bus off*).
- C2** Unstressed personal pronouns must precede the particle (e.g. *They ate it up* but not **They ate up it*).
- C3** If the particle precedes a simple definite NP, the particle does not take the NP as its object (e.g. in *He brought **along** his girlfriend*) unlike with PP complements or modifiers (e.g. in *He slept **in** the hotel*). This means that in the first example the NP is not a complement of the particle *along*, while in the second it is.

²The distinction between a VPC and a prepositional verb may be quite subtle, and as pointed out by Bolinger, many of the criteria proposed for diagnosing VPCs give different results for the same combination, frequently including unwanted combinations and excluding genuine VPCs.

In this paper we use the first two criteria, therefore the candidates may contain noise (in the form of prepositional verbs and related constructions).

VPCs have been the subject of a considerable amount of interest, and some analysis has been done on the subject of productive VPCs. In many cases the particle seems to be compositionally adding a specific meaning to the construction and following a productive pattern (e.g. in *tear up*, *cut up* and *split up*, where the verbs are semantically related and *up* adds a sense of completion to the action of these verbs). Fraser (1976) points out that semantic properties of verbs can affect their ability to combine with particles: for example, *bolt/cement/clamp/glue/paste/nail* are semantically similar verbs where the objects represented by the verbs are used to join material, and they can all combine with *down*. There is clearly a common semantic thread running through this list, so that a new verb that is semantically similar to them can also be reasonably assumed to combine with *down*. Indeed, frequently new VPCs are formed by analogy with existing ones, where often the verb is varied and the particle remains (e.g. *hang on*, *hold on* and *wait on*). Similarly, particles from a given semantic class can be replaced by other particles from the same class in compositional combinations: *send up/in/back/away* (Wurmbrand, 2000). By identifying classes of verbs that follow patterns such as these in VPCs, we can help in the identification of a new unknown candidate combination, using the degree of productivity of a class to which the verb belongs as a back-off strategy.

In terms of methods for automatic identification of VPCs from corpora, Baldwin (2005) proposes the extraction of VPCs with valence information from raw text, exploring a range of techniques (using (a) a POS tagger, (b) a chunker, (c) a chunk grammar, (d) a dependency parser, and (e) a combination of all methods). Villavicencio (2005) uses the Web as a corpus and productive patterns of combination to generate and validate candidate VPCs. The identification of compositionality in VPCs is addressed by McCarthy et al. (2003) who examine the overlap of similar words in an automatically acquired distributional thesaurus for verb and VPCs, and by Bannard (2005) who uses a distributional approach to determine when and to what extent the components of a VPC contribute their simplex meanings to the interpretation of the VPC. Both report a correlation between some of

the measures and compositionality judgements.

3 The Underlying Hypotheses

The problem of the automatic detection and classification of VPCs can be summarised as, for a given VPC candidate, to answer to the questions:

- Q1** Is it a real VPC or some free combination of verb and preposition/adverb or a prepositional verb?
- Q2** If it is a true VPC, is it idiomatic or compositional?

In order to answer the first question, we use two assumptions. Firstly, we consider that the elements of a true VPC co-occur above chance. The greater the correlation between the verb and the particle the greater the chance that the candidate is a true VPC. Secondly, based on criterion C1 we also assume that VPCs have more flexible syntax and are more productive than non-VPCs. This second assumption goes against what is usually adopted for general MWEs, since it is the prepositional verbs that allow less syntactic configurations than VPCs and are therefore more rigid (§ 2). To further distinguish VPCs from prepositional verbs and other related constructions we also verify the possibility of the particle to be immediately followed by an indirect prepositional complement (like in *The plane took off from London*), which is a good indicator/delimiter of a VPC since in non-VPC constructions like prepositional verbs the preposition needs to have an NP complement. Therefore, we will assume that a true VPC occurs in the following configurations, according to Villavicencio (2005) and Ramisch et al. (2008):

- S1** VERB + PARTICLE + DELIMITER, for intransitive VPCs;
- S2** VERB + NP + PARTICLE + DELIMITER, for transitive split VPCs and;
- S3** VERB + PARTICLE + NP + DELIMITER, for transitive joint VPCs.

In order to answer Q2, we look at the link between productivity and compositionality and assume that a compositional VPC accepts the substitution of one of its members by a semantically related term. This is in accordance to Fraser (1976), who shows that semantic properties of

verbs can affect their ability to combine with particles: for example verbs of hunting combining with the resultative *down* (*hunt/track/trail/follow down*) and verbs of cooking with the aspectual *up* (*bake/cook/fry/broil up*), forming essentially productive VPCs. Idiomatic VPCs, however, will not accept the substitution of one of its members by a related term (e.g. *get* and its synonyms in *get/*obtain/*receive over*), even if at first glance this could seem natural. In our experiments, we will consider that a VPC is compositional if it accepts: the replacement of the verb by a synonym, or of the preposition by another preposition. Summarising our hypothesis, we get:

- For Q1: Is the candidate syntactically flexible, i.e. does it allow the configurations S1 through S3?
 - NO: non-VPC
 - YES: VPC
- For Q2: Is the candidate semantically flexible, allowing the substitution of a member by a related word?
 - NO: idiomatic VPC
 - YES: compositional VPC

4 Data Sources

To generate a gold standard, we used the Baldwin VPC candidates dataset (henceforth Baldwin CD)³, which contains 3,078 English VPC candidates annotated with information about idiomaticity (14.5% are considered idiomatic). We further annotated this dataset with information about whether each candidate is a genuine VPC or not, where a candidate is considered *genuine* if it belongs to at least one of a set of machine-readable dictionaries: the Alvey Natural Language Tools (ANLT) lexicon (Carroll and Grover, 1989), the Comlex lexicon (Macleod and Grishman, 1998), and the LinGO English Resource Grammar (ERG) (Copestake and Flickinger, 2000)⁴. With this criterion 81.8% of them are considered genuine VPCs.

To gather information about the candidates in this work we employ both a fragment of 1.8M sentences from the British National Corpus (BNC Burnard (2000)) and the Web as corpora. The BNC fragment is used to calculate the correlation

³This dataset was provided by Timothy Baldwin for the MWE2008 Workshop.

⁴Version of November 2001.

measures since they require a corpus with known size. The Web is used to generate frequencies for the entropy measures, as discussed in § 5.2. Web frequencies are approximated by the number of pages containing a candidate and indexed by Yahoo Search API. In order to keep the searches as simple and self-sufficient as possible, no additional sources of information are used (Villavicencio, 2005). Therefore, the frequencies are quite conservative in the sense that by employing inflected forms of verbs, potentially much more evidence could be gathered.

For the generation of semantic variational patterns, we use both Wordnet 3.0 (Fellbaum, 1998) and Levin’s English Verb Classes and Alternations (Levin, 1993). Wordnet is organised as a graph of concepts, called *synsets*, linked by relations of synonymy, hyponymy, etc. Each synset contains a list of words that represent the concept. The verbs in a synset and its synonym synsets are used to generate variations of a VPC candidate. Likewise we use Levin’s classes, which define 190 fine-grained classes for English verbs, based on their syntactic and semantic features.

It is important to highlight that the generation of the semantic variations strongly relies on these resources. Therefore, cross-language extension would depend on the availability of similar tools for the target language.

5 Carrying out the experiments

Our experiments are composed of two stages, each one consisting of three steps (corresponding to the next three sections). The first stage filters out every candidate that is evaluated as not being a VPC, while the second one intends to identify the idiomatic VPCs among the remaining candidates of the previous stage.

5.1 Generating candidates

For each of the 3,078 items in the Baldwin CD we generated 2 sets of variations, syntactic and semantic, and we will refer to these as *alternative forms* or *variations* of a candidate.

The syntactic variations are generated using the patterns S1 to S3 described in section 3. Following the work of Villavicencio (2005) 3 frequently used prepositions *for*, *from* and *with* are used as delimiters and we search for NPs in the form of pronouns like *this* and definite NPs like *the boy*. The use of alternative search patterns also helps to give an in-

dication about the syntactic distribution of a candidate VPC, and consequently if it has a preferred syntactic realisation. For instance, for *eat up* and the delimiter *with*, we propose a list of Web search queries for its respective variations v_i , shown with their corresponding Web frequencies in table 1.⁵

Variation (v_i)	Frequency ($n_{Yahoo}(v_i)$)
eat up with	49200
eat the * up with	2240
eat this up with	1120
eat up the * with	3110

Table 1: Distribution of syntactic variations for the candidate *eat up*.

For the semantic variations, in order to capture the idiomaticity of VPCs we generate the alternative forms by replacing the verb by its synonym verbs as follows:

WNS Wordnet Strict variations. When using Wordnet, we consider any verb that belongs to the same synset of the candidate as a synonym.

WNL Wordnet Loose variations. This is an indirect synonymy relation capturing any verb in Wordnet that belongs either to the same synset or to a synset that is synonym of the synset in which the candidate verb is contained.

Levin These include all verbs in the same Levin class as the candidate.

Multiword synonyms are ignored in this step to avoid noisy search patterns, (e.g. **eat up up*). The examples for these variations are shown in table 2 for the candidate *act in*.

Wordnet and Levin are considered ambiguous resources because one verb is potentially contained in several synsets or classes. However, as Word Sense Disambiguation is not within the scope of this work we employ some heuristics to select a given sense for the candidate verb. In order to test the effect of frequency, the first heuristic adopts the first synset in the list, as Wordnet organises synsets in descending order of frequency (denoted as *first*). To study the influence of the number of synonyms, the second and third heuristics use respectively the biggest (*max*) and smallest (*min*) synsets. The last

⁵The Yahoo wildcard used in these searches matches any word occurring in that particular position.

Variation (v_i)	Source	$n_{Yahoo}(v_i)$
act in	—	2690
playact in	<i>WNS</i>	0
play in	<i>WNS</i>	167000
behave in	<i>WNL</i>	98
do in	<i>WNL</i>	24600
pose in	<i>Levin</i>	1610
qualify in	<i>Levin</i>	358
rank in	<i>Levin</i>	706
rate in	<i>Levin</i>	16700
serve in	<i>Levin</i>	2240

Table 2: Distribution of syntactic variations for the candidate *eat up*.

heuristic is the union of all synonyms (*all*). These heuristics are indicated using a subscript notation, where e.g. *WNS_{all}* symbolizes the WNS variations set using the union of all synsets as disambiguation heuristic. Finally, we generated two additional sets of candidates by replacing the particle by one of the 48 prepositions listed in the ANLT dictionary (*prep*) and also by one of 9 chosen locative prepositions (*loc-prep*). It is important to also verify possible variations of the preposition because compositional VPCs combine productively with one or more groups of particles, e.g. locatives, and present consequently a wider probability distribution among the variations, while an idiomatic VPC presents a higher frequency for a chosen preposition.

5.2 Working the statistical measures out

The classifications of the candidate VPCs are done using a set of measures: the frequencies of the VPC candidates and of their individual words, their Mutual Information (MI), χ^2 and Entropies. We calculate the MI and χ^2 indices of a candidate formed by a verb and a particle based on their individual frequencies and on their co-occurrence in the BNC fragment.

The *Entropy* measure is given by

$$H(V) = - \sum_{i=1}^n p(v_i) \ln [p(v_i)]$$

where

$$p(v_i) = \frac{n(v_i)}{\sum_{\forall v_j \in V} n(v_j)}$$

is the probability of the variation v_i to occur among the set of all possible variations $V =$

$H(V) \leq 0.001081$
$n_{BNC}(p) \leq 51611$
$n_{Yahoo}(v_{transitive}) \leq 1$
$n_{Yahoo}(v) \leq 2020000000 : yes$
$n_{Yahoo}(v) > 2020000000$
$\chi^2 \leq 25.99$
...

Figure 1: Fragment of the decision tree that filters out non-VPCs.

$\{v_1, v_2, \dots, v_n\}$, and $n(v_i)$ is the Web frequency for the variation v_i .

The entropy of a probability distribution gives us some clues about its shape. A very low entropy is a sign of a heterogeneous distribution that contains a peak. On the other hand, a distribution that presents uniformity will lead to a high entropy value.

The interest of $H(V)$ for the detection of VPCs is in that true instances are more likely to not prefer a canonical form, more widely distributing probabilities over all alternative syntactic frames (S1 to S3), while non-VPCs are more likely to choose one frame and present low frequencies for the proposed variations.

For the semantic variations, the entropy is calculated from a set V of variations generated by the Wordnet synset, Levin class and preposition substitutions described in § 5.1. The interpretation of the entropy at this point is that high entropy indicates compositionality while low entropy indicates idiomaticity, since compositional VPCs are more productive and distribute well over a class of verbs or a class of prepositions and idiomatic VPCs prefer a specific verb or preposition.

5.3 Bringing estimations together

Once we got a set of measures to predict VPCs and another to predict their idiomaticity/compositionality, we would like to know which measures are useful. Therefore, we combine our measures automatically by building a decision tree with the J48 algorithm, a version of the traditional entropy-based C4.5 algorithm implemented in the Weka package.⁶

6 Weighting the results up

The first stage of our experiments applied to the 3,078 VPC candidates generated a decision tree us-

ing 10-fold cross validation that is partially reproduced in figure 1. From these, 2,848 candidates were considered genuine VPCs, with 2,419 true positives, 100 false negatives and 429 false positives. This leads to a recall of 96% of the VPCs being kept in the list with a precision of 84.9%, and an f-measure of 90.1%. We interpret this as a very positive result since although some false negatives have been filtered out, the remaining candidates are now less noisy.

Figure 1 shows that the entropy of the variations is the best predictor since it is at the root of the tree. We can also see that there are several types of raw frequencies being used before a correlation measure appears (χ^2). We can conclude that the frequency of each transitive, intransitive and split configurations are also good predictors to detect false from true VPCs. At this point, MI does not seem to contribute to the classification task.

For our second stage, we generated Wordnet synonym, Levin class and preposition variations for a list of the 2,867 VPC candidates classified as genuine cases. We also took into account the proportion of synonyms that are MWEs (*vpc-syn*) and the proportion of synonyms that contain the candidate itself (*self-syn*).

In order to know what kind of contribution each measure gives to the construction of the decision tree, we used a simple iterative algorithm that constructs the set U of useful attributes. It first initialises U with all attributes, then calculates the precision for each class (yes and no)⁷ on a cross validation using all attributes in U . For each attribute $a \in U$, it ignores a and recalculates precisions. If both precisions decrease, the contribution of a is positive, if both increase then a is negative, else its contribution remains unknown. All features that contribute negatively are removed from U , and the algorithm is repeated until there is no negative attribute left.

The step-by-step execution of the algorithm can be observed in table 3, where the inconclusive steps are hidden. We found out that the optimal features are $U^* = \{self-syn, H(pre), H(Levin_{first}), H(WNS_{first}), H(WNS_{min}), H(Levin_{max}), H(Levin_{min})\}$. The *self-syn* information seems to be very important, as without it precisions of both classes decrease considerably

⁷We use the precision as a quality estimator since it gives a good idea of the amount of work that a grammar engineer or lexicographer must perform in order to clear the list from false positives.

⁶<http://www.cs.waikato.ac.nz/ml/weka/>

#	Ignored	Precision		
		No	Yes	+/-
1 st iteration				
0	—	86.6%	54.9%	
1	<i>vpc-syn</i>	86.7%	56.6%	—
2	<i>self-syn</i>	85.2%	28.7%	+
4	<i>H(loc-prep)</i>	86.7%	56.1%	—
6	<i>H(WNS_{max})</i>	87.5%	57.4%	—
9	<i>H(WNL_{first})</i>	86.7%	57.9%	—
10	<i>H(WNL_{max})</i>	86.7%	57.8%	—
11	<i>H(WNL_{min})</i>	86.9%	57.6%	—
16	<i>H(Levin_{all})</i>	86.7%	55.1%	—
2 nd iteration				
17	—	87.7%	60.3%	
18	<i>H(prepare)</i>	87.6%	59.2%	+
21	<i>H(WNS_{all})</i>	87.8%	61.6%	—
22	<i>H(WNL_{all})</i>	87.8%	61.0%	—
23	<i>H(Levin_{first})</i>	87.5%	60.2%	+
3 rd iteration				
26	—	87.8%	61.9%	
27	<i>H(WNS_{first})</i>	87.8%	61.9%	±
28	<i>H(WNS_{min})</i>	87.7%	61.1%	+
29	<i>H(Levin_{max})</i>	87.8%	61.6%	±
30	<i>H(Levin_{min})</i>	87.7%	61.5%	+

Table 3: Iterative attributes selection process. Precision in each class is used as quality estimator.

(experiment #2).

All entropies of the WNL heuristics are of little or no utility. This could probably be explained by either the choice of simple WSD heuristics for selecting synsets, or because the indirect synonymy information is too far related to the original verb to be used in variational patterns. Inspecting the generated variations, we notice that most of the synonym synsets are related to secondary senses or very specific uses of a verb and are thus not correctly disambiguated.

In what concerns the WNS sets, only the smallest and first synset were kept, suggesting again that it may not be a good idea to maximise the synonyms set and for future work, we intent to establish a threshold for a synset to be taken into account. In addition, we can also infer a positive contribution of the frequency of a sense with the choice of the first synset returned by Wordnet resulting in a reasonable WSD heuristic (which is compatible with the results by McCarthy et al. (2004)).

On the other hand, the algorithm selected the

first, the smallest and the biggest of the Levin’s sets. This probably happens because the majority of these verbs belongs only to one or two, but never to a great number of classes. Since the granularity of the classes is coarser than for synsets, the heuristics often offer four equal or very close entropies and thus redundant information. As an overall result, the last iteration shown in table 3 indicates a precision of 61.9% for the classifier in detecting idiomatic VPCs, that is to say that we automatically retrieved 176 VPCs where 67 are false positives and 109 are truly idiomatic. This value is a quality estimator for the resulting VPCs that will potentially be used in the construction of a lexicon. Recall of idiomatic VPCs goes from 16.7% to 24.9%.

7 Conclusions

One of the important challenges for robust natural language processing systems is to be able to successfully deal with Multiword Expressions and related constructions. We investigated the identification of VPCs using a combination of statistical methods and linguistic information, and whether there is a correlation between the productivity of VPCs and their semantics that could help us detect if a VPC is idiomatic or compositional.

The results confirm that the use of statistical and linguistic information to automatically identify verb-particle constructions presents a reasonable way of improving coverage of existing lexical resources in a very simple and straightforward manner. In terms of grammar engineering, the information about compositional candidates belonging to productive classes provides us with the basis for constructing a family of fine-grained redundancy rules for these classes. These rules are applied in a constrained way to verbs already in the lexicon, according to their semantic classes. The VPCs identified as idiomatic, on the other hand, need to be explicitly added to the lexicon, after their semantic is determined. This study can also be complemented with the results of investigations into the semantics of VPCs, as discussed by both Bannard (2005) and McCarthy et al. (2003).

In addition, the use of clustering methods is an interesting possibility for automatically identifying clusters of productive classes of both verbs and of particles that combine well together.

Acknowledgments

This research was partly supported by the CNPq research project *Recuperação de Informações Multilíngües* (CNPq Universal 484585/2007-0).

References

- Baldwin, Timothy, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2004. Road-testing the English Resource Grammar over the British National Corpus. In *Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal.
- Baldwin, Timothy. 2005. Deep lexical acquisition of verb-particle constructions. *Computer Speech and Language*, 19(4):398–414.
- Bannard, Colin J. 2005. Learning about the meaning of verb-particle constructions from corpora. *Computer Speech and Language*, 19(4):467–478.
- Bolinger, Dwight. 1971. *The phrasal verb in English*. Harvard University Press, Harvard, USA.
- Burnard, Lou. 2000. User reference guide for the British National Corpus. Technical report, Oxford University Computing Services.
- Carroll, John and Claire Grover. 1989. The derivation of a large computational lexicon of English from LDOCE. In Boguraev, B. and E. Briscoe, editors, *Computational Lexicography for Natural Language Processing*. Longman.
- Copestake, Ann and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000)*.
- Dehé, Nicole. 2002. *Particle verbs in English: syntax, information structure and intonation*. John Benjamins, Amsterdam/Philadelphia.
- Evert, Stefan and Brigitte Krenn. 2005. Using small random samples for the manual evaluation of statistical association measures. *Computer Speech and Language*, 19(4):450–466.
- Fazly, Afsaneh and Suzanne Stevenson. 2006. Automatically constructing a lexicon of verb phrase idiomatic combinations. In *EACL*. The Association for Computer Linguistics.
- Fellbaum, Christiane, editor. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May.
- Fraser, Bruce. 1976. *The Verb-Particle Combination in English*. Academic Press, New York, USA.
- Jackendoff, Ray. 2002. English particle constructions, the lexicon, and the autonomy of syntax. In N. Dehé, R. Jackendoff, A. McIntyre and S. Urban, editors, *Verb-Particle Explorations*. Berlin: Mouton de Gruyter.
- Levin, Beth. 1993. *English Verb Classes and Alternations: a preliminary investigation*. University of Chicago Press, Chicago and London.
- Macleod, Catherine and Ralph Grishman. 1998. Complex syntax reference manual, Proteus Project.
- McCarthy, Diana, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL 2003 workshop on Multiword expressions*, pages 73–80, Morristown, NJ, USA. Association for Computational Linguistics.
- McCarthy, Diana, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 279. Association for Computational Linguistics.
- Pearce, Darren. 2002. A comparative evaluation of collocation extraction techniques. In *Third International Conference on Language Resources and Evaluation*, Las Palmas, Canary Islands, Spain.
- Ramisch, Carlos, Paulo Schreiner, Marco Idiart, and Aline Villavicencio. 2008. An evaluation of methods for the extraction of multiword expressions. In *Proceedings of the LREC Workshop - Towards a Shared Task for Multiword Expressions (MWE 2008)*, pages 50–53, Marrakech, Morocco, June.
- Sharoff, Serge. 2004. What is at stake: a case study of russian expressions starting with a preposition. pages 17–23, Barcelona, Spain.
- Villavicencio, Aline. 2005. The availability of verb-particle constructions in lexical resources: How much is enough? *Journal of Computer Speech and Language Processing*, 19(4):415–432.
- Wurmbrand, S. 2000. The structure(s) of particle verbs. Ms., McGill University.
- Zhang, Yi, Valia Kordoni, Aline Villavicencio, and Marco Idiart. 2006. Automated multiword expression prediction for grammar engineering. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 36–44, Sydney, Australia. Association for Computational Linguistics.

Fast Mapping in Word Learning: What Probabilities Tell Us

Afra Alishahi and Afsaneh Fazly and Suzanne Stevenson

Department of Computer Science

University of Toronto

{afra,afsaneh,suzanne}@cs.toronto.edu

Abstract

Children can determine the meaning of a new word from hearing it used in a familiar context—an ability often referred to as *fast mapping*. In this paper, we study fast mapping in the context of a general probabilistic model of word learning. We use our model to simulate fast mapping experiments on children, such as referent selection and retention. The word learning model can perform these tasks through an inductive interpretation of the acquired probabilities. Our results suggest that fast mapping occurs as a natural consequence of learning more words, and provides explanations for the (occasionally contradictory) child experimental data.

1 Fast Mapping

An average six-year-old child knows over 14,000 words, most of which s/he has learned from hearing other people use them in ambiguous contexts (Carey, 1978). Children are thus assumed to be equipped with powerful mechanisms for performing such a complex task so efficiently. One interesting ability children as young as two years of age show is that of correctly and immediately mapping a novel word to a novel object in the presence of other familiar objects. The term “fast mapping” was first used by Carey and Bartlett (1978) to refer to this phenomenon.

Carey and Bartlett’s goal was to examine how much children learn about a word when presented in an ambiguous context, as opposed to concentrated teaching. They used an unfamiliar name (*chromium*) to refer to an unfamiliar color (olive green), and then asked a group of four-year-old children to select an object from among a set, upon hearing a sentence explicitly

asking for the object of the new color, as in: *bring the chromium tray, not the blue one*. Children were generally good at performing this “referent selection” task. In a production task performed six weeks later, when children had to use the name of the new color, they showed signs of having learned something about the new color name, but were not successful at producing it. On the basis of these findings, Carey and Bartlett suggest that fast mapping and word learning are two distinct, yet related, processes.

Extending Carey and Bartlett’s work, much research has concentrated on providing an explanation for fast mapping, and on examining its role in word learning. These studies also show that children are generally good at referent selection, given a novel target. However, there is not consistent evidence regarding whether children actually learn the novel word from one or a few such exposures (retention). For example, whereas the children in the experiments of Golinkoff et al. (1992) and Halberda (2006) showed signs of nearly-perfect retention of the fast-mapped words, those in the studies reported by Horst and Samuelson (2008) did not (all participating children were close in age range).

There are also many speculations about the possible causes of fast mapping. Some researchers consider it as a sign of a specialized (innate) mechanism for word learning. Markman and Wachtel (1988), for example, argue that children fast map because they expect each object to have only one name (mutual exclusivity). Golinkoff et al. (1992) attribute fast mapping to a (hard-coded) bias towards mapping novel names to nameless object categories. Some even suggest a change in children’s learning mechanisms, at around the time they start to show evidence of fast mapping (which coincides with a sudden burst in their vocabulary), e.g., from associative to referential (Gopnik and Meltzoff, 1987; Reznick and Goldfield, 1992). In contrast, others see fast mapping as a phenomenon that arises from more general processes of learning

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

and/or communication, which also underlie the impressive rate of lexical acquisition in children (e.g., Clark, 1990; Diesendruck and Markson, 2001; Regier, 2005; Horst et al., 2006; Halberda, 2006).

In our previous work (Fazly et al., 2008), we presented a word learning model which proposes a probabilistic interpretation of cross-situational learning, and bootstraps its own partially-learned knowledge of the word meanings to accelerate word learning over time. We have shown that the model can learn reasonable word–meaning associations from child-directed data, and that it accounts for observed learning patterns in children, such as vocabulary spurt, without requiring a developmental change in the underlying learning mechanism. Here, we use this computational model to investigate fast mapping and its relation to word learning. Specifically, we take a close look at the onset of fast mapping in our model by simulating some of the psychological experiments mentioned above. We examine the behaviour of the model in various referent selection and retention tasks, and provide explanations for the (occasionally contradictory) experimental results reported in the literature. We also study the effect of exposure to more input on the performance of the model in fast mapping.

Our results suggest that fast mapping can be explained as an induction process over the acquired associations between words and meanings. Our model learns these associations in the form of probabilities within a unified framework; however, we argue that different interpretations of such probabilities may be involved in choosing the referent of a familiar as opposed to a novel target word (as noted by Halberda, 2006). Moreover, the overall behaviour of our model confirms that the probabilistic bootstrapping approach to word learning naturally leads to the onset of fast mapping in the course of lexical development, without hard-coding any specialized learning mechanism into the model to account for this phenomenon.

2 Overview of the Computational Model

This section summarizes the model presented in Fazly et al. (2008). Our word learning algorithm is an adaptation of the IBM translation model proposed by Brown et al. (1993). However, our model is incremental, and does not require a batch process over the entire data.

2.1 Utterance and Meaning Representations

The input to our word learning model consists of a set of utterance–scene pairs that link an observed scene (what the child perceives) to the utterance that describes it (what the child hears). We represent each utterance as a sequence of words, and the correspond-

ing scene as a set of meaning symbols. To simulate *referential uncertainty* (i.e., the case where the child perceives aspects of the scene that are unrelated to the perceived utterance), we include additional symbols in the representation of the scene, e.g.:

Utterance: *Joe rolled the ball*

Scene: {joe, roll, the, ball, mommy, hand, talk}

In Section 3.1, we explain how the utterances and the corresponding semantic symbols are selected, and how we add referential uncertainty.

Given a corpus of such utterance–scene pairs, our model learns the meaning of each word w as a probability distribution, $p(\cdot|w)$, over the semantic symbols appearing in the corpus. In this representation, $p(m|w)$ is the probability of a symbol m being the meaning of a word w . In the absence of any prior knowledge, all symbols are equally likely to be the meaning of a word. Hence, prior to receiving any usages of a given word, the model assumes a uniform distribution over semantic symbols as its meaning.

2.2 Meaning Probabilities

Our model combines probabilistic interpretations of cross-situational learning (Quine, 1960) and of a variation of the principle of contrast (Clark, 1990), through an interaction between two types of probabilistic knowledge acquired and refined over time. Given an utterance–scene pair received at time t , i.e., $(U^{(t)}, S^{(t)})$, the model first calculates an alignment probability a for each $w \in U^{(t)}$ and each $m \in S^{(t)}$, using the meaning probabilities $p(\cdot|w)$ of all the words in the utterance prior to this time. The model then revises the meaning of the words in $U^{(t)}$ by incorporating the alignment probabilities for the current input pair. This process is repeated for all the input pairs, one at a time.

Step 1: Calculating the alignment probabilities.

We estimate the alignment probabilities of words and meaning symbols based on a localized version of the principle of contrast: that a meaning symbol in a scene is likely to be highly associated with only one of the words *in the corresponding utterance*.¹ For a symbol $m \in S^{(t)}$ and a word $w \in U^{(t)}$, the higher the probability of m being the meaning of w (according to $p(m|w)$), the more likely it is that m is aligned with w in the current input. In other words, $a(w|m, U^{(t)}, S^{(t)})$ is proportional to $p^{(t-1)}(m|w)$. In addition, if there is strong evidence that m is the meaning of another word in $U^{(t)}$ —i.e., if $p^{(t-1)}(m|w')$ is high for some $w' \in U^{(t)}$ other

¹Note that this differs from what is widely known as the principle of contrast (Clark, 1990), in that the latter assumes contrast across the entire vocabulary rather than within an utterance.

than w —the likelihood of aligning m to w should decrease. Combining these two requirements:

$$a(w|m, U^{(t)}, S^{(t)}) = \frac{p^{(t-1)}(m|w)}{\sum_{w' \in U^{(t)}} p^{(t-1)}(m|w')} \quad (1)$$

Due to referential uncertainty, some of the meaning symbols in the scene might not have a counterpart in the utterance. To accommodate for such cases, a dummy word is added to each utterance before the alignment probabilities are calculated, in order to let a meaning symbol not be (strongly) aligned with any of the words in the current utterance.

Step 2: Updating the word meanings. We need to update the probabilities $p(\cdot|w)$ for all words $w \in U^{(t)}$, based on the evidence from the current input pair reflected in the alignment probabilities. We thus add the current alignment probabilities for w and the symbols $m \in S^{(t)}$ to the accumulated evidence from prior co-occurrences of w and m . We summarize this cross-situational evidence in the form of an association score, which is updated incrementally:

$$\text{assoc}^{(t)}(w, m) = \text{assoc}^{(t-1)}(w, m) + a(w|m, U^{(t)}, S^{(t)}) \quad (2)$$

where $\text{assoc}^{(t-1)}(w, m)$ is zero if w and m have not co-occurred before. The association score of a word and a symbol is basically a weighted sum of their co-occurrence counts.

The model then uses these association scores to update the meaning of the words in the current input:

$$p^{(t)}(m|w) = \frac{\text{assoc}^{(t)}(m, w) + \lambda}{\sum_{m_j \in \mathcal{M}} \text{assoc}^{(t)}(m_j, w) + \beta \times \lambda} \quad (3)$$

where \mathcal{M} is the set of all symbols encountered prior to or at time t , β is the expected number of symbol types, and λ is a small smoothing factor. The denominator is a normalization factor to get valid probabilities. This formulation results in a uniform probability of $1/\beta$ over all $m \in \mathcal{M}$ for a novel word w , and a probability smaller than λ for a meaning symbol m that has not been previously seen with a familiar word w .

Our model updates the meaning of a word every time it is heard in an utterance. The strength of learning of a word at time t is reflected in $p^{(t)}(m = m_w|w)$, where m_w is the ‘‘correct’’ meaning of w : for a learned word w , the probability distribution $p(\cdot|w)$ is highly skewed towards the correct meaning m_w , and therefore hearing w will trigger the retrieval of the meaning m_w .²

²An input-generation lexicon contains the correct meaning for each word, as described in Section 3.1. Note that the model does not have access to this lexicon for learning; it is used only for input generation and evaluation.

From this point on, we simply use $p(m|w)$ (omitting the superscript (t)) to refer to the meaning probability of m for w at the present time of learning.

2.3 Referent Probabilities

The meaning probability $p(m|w)$ is used to retrieve the most probable meaning for w among all the possible meaning symbols m . However, in the referent selection tasks performed by children, the subject is often forced to select the referent of a target word from among a limited set of objects, even when the meaning of the target word has not been accurately learned yet. For our model to perform such tasks, it has to decide how likely it is for a target word w to refer to a particular object m , based on its previous knowledge about the mapping between m and w (i.e., $p(m|w)$), as well as the mapping between m and other words in the lexicon.³

The likelihood of using a particular name w to refer to a given object m is calculated as:

$$\begin{aligned} rf(w|m) &= p(w|m) \\ &= \frac{p(m|w) \cdot p(w)}{p(m)} \\ &= \frac{p(m|w) \cdot p(w)}{\sum_{w' \in \mathcal{V}} p(m|w') \cdot p(w')} \end{aligned} \quad (4)$$

where \mathcal{V} is the set of all words that the model has seen so far, and $p(w)$ is the relative frequency of w :

$$p(w) = \frac{\text{freq}(w)}{\sum_{w' \in \mathcal{V}} \text{freq}(w')} \quad (5)$$

The referent of a target word w among the present objects, therefore, will be the object m with the highest referent probability $rf(w|m)$.

3 Experimental Setup

3.1 The Input Corpora

We extract utterances from the Manchester corpus (Theakston et al., 2001) in the CHILDES database (MacWhinney, 2000). This corpus contains transcripts of conversations with children between the ages of 1;8 and 3;0 (years;months). We use the mother’s speech from transcripts of 6 children, remove punctuation and lemmatize the words, and concatenate the corresponding sessions as input data.

There is no semantic representation of the corresponding scenes available from CHILDES. Therefore, we automatically construct a scene representation for each utterance, as a set containing the semantic referents of the words in that utterance. We get these from an input-generation lexicon that contains a symbol associated with each word as its semantic

³All through the paper, we use m as both the meaning and the referent of a word w .

referent. We use every other sentence from the original corpus, preserving their chronological order. To simulate referential uncertainty in the input, we then pair each sentence with its own scene representation as well as that of the following sentence in the original corpus. (Note that the latter sentence is not used as an utterance in our input.) The extra semantic symbols that are added to each utterance thus correspond to meaningful semantic representations, as opposed to randomly selected symbols. In the resulting corpus of 92,239 input pairs, each utterance is, on average, paired with 78% extra meaning symbols, reflecting a high degree of referential uncertainty.

3.2 The Model Parameters

We set the parameters of our learning algorithm using a development data set which is similar to our training and test data, but is selected from a non-overlapping portion of the Manchester corpus. The expected number of symbols, β in Eq. (3), is set to 8500 based on the total number of distinct symbols extracted for the development data. Therefore, the default probability of a symbol for a novel word will be $1/8500$. A familiar word, on the other hand, has been seen with some symbols before. Therefore, the probability of a previously unseen symbol for it (which, based on Eq. (3), has an upper bound of λ) must be less than the default probability mentioned above. Accordingly, we set λ to 10^{-5} .

3.3 The Training Procedure

In the next section, we report results from the computational simulation of our model for a number of experiments. All of the simulations use the same parameter settings (as described in the previous section), but different input: in each simulation, a random portion of 1000 utterance–scene pairs is selected from the input corpus, and incrementally processed by the model. The size of the training corpus is chosen arbitrarily to reflect a sample point in learning, and further experiments have shown that increasing this number does not change the pattern observed in the results. In order to avoid behaviour that is specific to a particular sequence of input items, the reported results in the next section are averaged over 10 simulations.

4 Experimental Results and Analysis

4.1 Referent Selection

In a typical word learning scenario, the child faces a scene where a number of familiar and unfamiliar objects are present. The child then hears a sentence, which describes (some part of) the scene, and is composed of familiar and novel words (e.g., hearing *Joe is*

eating a cheem, where *cheem* is a previously unseen fruit). In such a setting, our model aligns the objects in the scene with the words in the utterance based on its acquired knowledge of word meanings, and then updates the meanings of the words accordingly. The model can align a familiar word with its referent with high confidence, since the previously learned meaning probability of the familiar object given the familiar word, or $p(m|w)$, is much higher than the meaning probability of the same object given any other word in the sentence. In a similar fashion, the model can easily align a novel word in the sentence with a novel object in the scene, because the meaning probability of the novel object given the novel word ($1/\beta$, according to Eq. (3)) is higher than the meaning probability of that object for any previously heard word in the sentence (the latter probability is smaller than λ in Eq. (3), as explained in Section 3.2).

Earlier fast mapping experiments on children assumed that it is such a contrast between the familiar and novel words in the same sentence that helps children select the correct target object in a referent selection task. For example, in Carey and Bartlett’s (1978) experiment, to introduce a novel word–meaning association (e.g., *chromium–olive*), the authors use both the familiar and the novel words in one sentence (*bring me the chromium tray, not the blue one.*). However, further experiments show that children can successfully select the correct referent even if such a contrast is not present in the sentence. Many researchers have performed experiments where young subjects are forced to choose between a novel and a familiar object upon hearing a request, such as *give me the ball* (familiar target), or *give me the dax* (novel target). In all of the reported experimental results, children can readily pick the correct referent for a familiar or a novel target word in such a setting (Golinkoff et al., 1992; Halberda and Goldman, 2008; Halberda, 2006; Horst and Samuelson, 2008).

However, Halberda’s eye-tracking experiments on both adults and pre-schoolers suggest that the processes involved for referent selection in the familiar target situation may be different from those in the novel target situation. In the latter situation, subjects appear to systematically reject the familiar object as the referent of the novel name before mapping the novel object to the novel name. In the familiar target situation, however, there is no need to reject the novel distractor object, because the subject already knows the referent of the target.

The difference between these two conditions can be explained in terms of the meaning and referent probabilities of our model explained in Section 2. In a typical referent selection experiment, the child is asked to

“*get the ball*” while facing a `ball` and a novel object (`dax`). We assume that the child knows the meaning of verbs and determiners such as *get* and *the*, therefore we simplify the familiar target condition in the form of the following input item:

ball (FAMILIAR TARGET)
`{ball, dax}`

A familiar word such as *ball* has a meaning probability highly skewed towards its correct meaning. That is, upon hearing *ball*, the model can confidently retrieve its meaning `ball`, which is the one with the highest probability $p(m|ball)$ among all possible meanings m . In such a case, if `ball` is present in the scene, the model can easily pick it as the referent of the familiar target name, without processing the other objects in the scene.

Now consider the condition where a novel target name is used in the presence of a familiar and a previously unseen object:

dax (NOVEL TARGET)
`{ball, dax}`

Since this is the first time the model has heard the word *dax*, both meanings `ball` and `dax` are equally likely because $p(.|dax)$ is uniform. Thus the meaning probabilities cannot be solely used for selecting the referent of *dax*, and the model has to perform some kind of induction on the potential referents in the scene based on what it has learned about each of them. The model can infer the referent of *dax* by comparing the referent probabilities $rf(dax|ball)$ and $rf(dax|dax)$ from Eq. (4) after processing the input item. Since `ball` has strong associations with another word *ball*, its referent probability for the novel name *dax* is much lower than the referent probability of `dax`, which does not have strong associations with any of the words in the learned lexicon.

We simulate the process of referent selection in our model as follows. We train the model as described in Section 3.3. We then present the model with one more input item, which represents either the FAMILIAR TARGET or the NOVEL TARGET condition. For each condition, we compare the meaning probability $p(object|target)$ for both familiar and novel objects in the scene (see Table 1, top panel). In the FAMILIAR TARGET condition, the model demonstrates a strong preference towards choosing the familiar object as the referent, whereas in the NOVEL TARGET condition, the model shows no preference towards any of the objects based on the meaning probabilities of the target word. Therefore, for the NOVEL TARGET condition, we also compare the referent probabilities $rf(target|object)$ for both objects after processing

Table 1: Referent selection in FAMILIAR and NOVEL TARGET conditions.

UPON HEARING THE TARGET WORD		
Condition	$p(ball target)$	$p(dax target)$
FAMILIAR TARGET	0.843 \pm 0.056	\ll 0.0001
NOVEL TARGET	0.0001 \pm 0.00	0.0001 \pm 0.00

AFTER PERFORMING INDUCTION		
Condition	$rf(target ball)$	$rf(target dax)$
NOVEL TARGET	0.127 \pm 0.127	0.993 \pm 0.002

the input item as a training pair, simulating the induction process that humans go through to select the referent in such cases. This time, the model shows a strong preference towards the novel object as the referent of the target word (see Table 1, bottom panel). Our results confirm that in both conditions, the model consistently selects the correct referent for the target word across all the simulations.

4.2 Retention

As discussed in the previous section, results from the human experiments as well as our computational simulations show that the referent of a novel target word can be selected based on the previous knowledge about the present objects and their names. However, the success of a subject in a referent selection task does not necessarily mean that the child/model has *learned* the meaning of the novel word based on that one trial. In order to better understand what and how much children learn about a novel word from a single ambiguous exposure, some studies have performed retention trials after the referent selection experiments. Often, various referent selection trials are performed in one session, where in each trial a novel object–name pair is introduced among familiar objects. Some of the recently introduced objects are then put together in one last trial, and the subjects are asked to choose the correct referent for one of the (recently heard) novel target words. The majority of the reported experiments show that children can successfully perform the retention task (Golinkoff et al., 1992; Halberda and Goldman, 2008; Halberda, 2006).

We simulate a similar retention experiment by training the model as usual. We further present the model with two experimental training items similar to the one used in the NOVEL TARGET condition in the previous section, with different familiar and novel objects and words in each input:

dax (REFERENT SELECTION TRIAL 1)
`{ball, dax}`

cheem (REFERENT SELECTION TRIAL 2)
`{pen, cheem}`

Table 2: Retention of a novel target word from a set of novel objects.

2-OBJECT RETENTION TRIAL		
$rf(dax dax)$	$rf(dax cheem)$	
0.996 ± 0.001	0.501 ± 0.068	

3-OBJECT RETENTION TRIAL		
$rf(dax dax)$	$rf(dax cheem)$	$rf(dax lukk)$
0.995 ± 0.001	0.407 ± 0.062	0.990 ± 0.001

The training session is followed by a retention trial, where the two novel objects used in the previous experimental inputs are paired with one of the novel target words:

dax (2-OBJECT RETENTION TRIAL)
 {cheem, dax}

After processing the retention input, we compare the referent probabilities $rf(dax|cheem)$ and $rf(dax|dax)$ to see if the model can choose the correct novel object in response to the target word *dax*. The top panel in Table 2 summarizes the results of this experiment. The model consistently shows a strong preference towards the correct novel object as the referent of the novel target word across all simulations.

Unlike studies on referent selection, experimental results for retention have not been consistent across various studies. Horst and Samuelson (2008) perform experiments with two-year-old children involving both referent selection and retention, and report that their subjects perform very poorly at the retention task. One factor that discriminates the experimental setup of Horst and Samuelson from others (e.g., Halberda, 2006) is that, in their retention trials, they put together two recently observed novel objects with a third novel object that has not been seen in any of the experimental sessions before. The authors do not attribute their contradictory results to the presence of this third object, but this factor can in fact affect the performance considerably. We simulate this condition by using the same input items for referent selection trials as in the previous simulation, but we replace the retention trial with the following:

dax (3-OBJECT RETENTION TRIAL)
 {cheem, dax, lukk}

The third object, *lukk*, has not been seen by the model before. Results under the new condition are reported in the bottom panel of Table 2. As can be seen, the model shows a strong tendency towards the correct novel referent *dax* for the novel target *dax*, compared to the other recently seen novel object *cheem*. However, the probability of the unseen object *lukk* is also very high for the target word *dax*. That is because the model cannot use any previously acquired

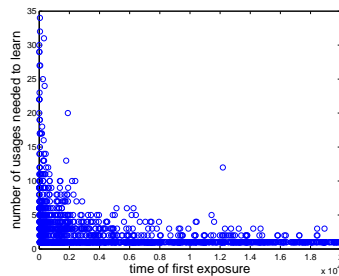


Figure 1: Number of usages needed to learn a word, as a function of the word’s age of exposure.

knowledge about *lukk* (i.e., associating it with another word) to rule it out as a referent for *dax*. These results show that introducing a new object for the first time in a retention trial considerably increases the difficulty of the task. This can explain the contradictory results reported in the literature: when the referent probabilities are not informative, other factors may influence the outcome of the experiment, such as the amount of training received for a novel word–object, or a possible delay between training and test sessions.

4.3 The Effect of Exposure to More Input

The fast mapping ability observed in children implies that once children have learned a repository of words, they can easily link novel words to novel objects in a familiar context based only on a few exposures. We examine this effect in our model: we train the model on 20,000 input pairs, looking at the relation between the time of first exposure to a word, and the number of usages that the model needs for learning that word. Figure 1 plots this for words that have been learned at some point during the training.⁴ We can see that the model shows clear fast mapping behaviour—that is, words received later in time, on average, require fewer usages to be learned. These results show that our model exhibits fast mapping patterns once it has been exposed to enough word usages, and that no change in the underlying learning mechanism is needed.⁵

The effect of exposure to more input on fast mapping can be described in terms of context familiarity: the more input the model has processed so far, the more likely it is that the context of the usage of a novel word (the other words in the sentence and the objects in the scene) is familiar to the model. This pattern has been studied through a number of experiments on

⁴We consider a word w as learned if the meaning probability $p(m_w|w)$ is higher than a certain threshold θ . For this experiment, we set $\theta = 0.70$.

⁵In Fazly et al. (2008), we reported a variation of this experiment, where we used a smaller training set, and also a different semantic representation for word meanings.

children. For example, Gershkoff-Stowe and Hahn (2007) taught 16- to 18-month-olds the names of 24 unfamiliar objects over 12 training sessions, where unfamiliar objects were presented with varying frequency. Data were compared to a control group of children who were exposed to the same experimental words at the first and last sessions only. Their results show that for children in the experimental group, extended practice with a novel set of words led to the rapid acquisition of a second set of low-practice words. Children in the control group did not show the same lexical advantage.

Inspired by Gershkoff-Stowe and Hahn (2007), we perform an experiment to study the effect of context familiarity on fast mapping in our model. We choose two sets of words, CONTEXT (containing 20 words) and TARGET (containing 10 words), to conduct a referent selection task as follows. First, we train our model on a sequence of utterance–scene pairs constructed from the set $\text{CONTEXT} \cup \text{TARGET}$, as follows: the unified set is randomly shuffled and divided into two subsets, words in each subset are put together to form an utterance, and the meanings of the words in that utterance are put together to form the corresponding scene. We repeat this process twice, so that each word appears in exactly two input pairs. We train our model on the constructed pairs.⁶ Next, we perform a referent selection task on each word in the TARGET set: we pair each target word w with the meaning of 10 randomly selected words from $\text{CONTEXT} \cup \text{TARGET}$, including the meaning of the target word itself (m_w), and have the model process this test pair. We compare the referent probability of w and each $m \in \text{CONTEXT} \cup \text{TARGET}$ to see whether the model can correctly map the target word to its referent. We call this setting the LOW TRAINING condition.

In the above setting, the context words in the referent selection trials are as new to the model as the target words. We thus repeat this experiment with a familiar context: we first train the model over input pairs that are randomly constructed from words in CONTEXT only, using the same training procedure as described above. This context-familiarization process is followed by a similar training session on $\text{CONTEXT} \cup \text{TARGET}$, and a test session on target words, similar to the previous condition. Again, we count the number of correct mappings between a target word and its referent based on the referent probabilities. We call this setting the HIGH TRAINING condition. Table 3 shows the results for both conditions. It can be seen that the accuracy of finding the referent

⁶Unlike in previous experiments, here we do not use child-directed data as we want to control the familiarity of the context.

Table 3: Average number of correct mappings and the referent probabilities of target words for two conditions, LOW and HIGH TRAINING.

Condition	Correct mappings	$P(\text{target} m_{\text{target}})$
LOW TRAINING	%54	0.216±0.04
HIGH TRAINING	%90	0.494±0.79

for a target word, as well as the referent probability of a target word for its correct meaning, increase as a result of more training on the context. In other words, a more familiar context helps the model perform better in a fast mapping task.

5 Related Computational Models

The rule-based model of Siskind (1996), and the connectionist model proposed by Regier (2005), both show that learning gets easier as the model is exposed to more input—that is, words heard later are learned faster. These findings confirm that fast mapping may simply be a result of learning more words, and that no explicit change in the underlying learning mechanism is needed. However, these studies do not examine various aspects of fast mapping, such as referent selection and retention. Horst et al. (2006) explicitly test fast mapping in their connectionist model of word learning by performing referent selection and retention tasks. The behaviour of their model matches the child experimental data reported in a study by the same authors (Horst and Samuelson, 2008), but not that of the contradictory findings of other similar experiments. Moreover, the model’s learning capacity is limited, and the fast mapping experiments are performed on a very small vocabulary. Frank et al. (2007) examine fast mapping in their Bayesian model by testing its performance in a novel target referent selection task. However, the experiment is performed on an artificial corpus. Moreover, since the learning algorithm is non-incremental, the success of the model in referent selection is determined implicitly: each possible word–meaning mapping from the test input is added to the current lexicon, and the consistency of the new lexicon is checked against the training corpus.

6 Discussion and Concluding Remarks

We have used a general computational model of word learning (first introduced in Fazly et al., 2008) to study fast mapping. Our model learns a probabilistic association between a word and its meaning, from exposure to word usages in naturalistic contexts. We have shown that these probabilities can be used to simulate various fast mapping experiments performed on children, such as referent selection and retention. Our

experimental results suggest that fast mapping can be explained as an induction process over the acquired associations between words and objects. In that sense, fast mapping is a general cognitive ability, and not a hard-coded, specialized mechanism of word learning.⁷ In addition, our results confirm that the onset of fast mapping is a natural consequence of learning more words, which in turn accelerates the learning of new words. This bootstrapping approach results in a rapid pace of vocabulary acquisition in children, without requiring a developmental change in the underlying learning mechanism.

Results of the referent selection experiments show that our model can successfully find the referent of a novel target word in a familiar context. Moreover, our retention experiments show that the model can map a recently heard novel word to its recently seen novel referent (among other novel objects) after only one exposure. However, the strength of the association of a novel pair after one exposure shows a notable difference compared to the association between a “typical” familiar word and its meaning.⁸ This is consistent with what is commonly assumed in the literature: even though children learn something about a word from only one exposure, they often need more exposure to reliably learn its meaning (Carey, 1978). Various kinds of experiments have been performed to examine how strongly children learn novel words introduced to them in experimental settings. For example, children are persuaded to produce a fast-mapped word, or to use the novel word to refer to objects that are from the same category as its original referent (e.g., Golinkoff et al., 1992; Horst and Samuelson, 2008). We intend to look at these new tasks in our future research.

References

- Behrend, Douglas A., Jason Scofield, and Erica E. Kleinknecht 2001. Beyond fast mapping: Young children’s extensions of novel words and novel facts. *Developmental Psychology*, 37(5):698–705.
- Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.
- Carey, Susan 1978. The child as word learner. In Halle, M., J. Bresnan, and G. A. Miller, editors, *Linguistic Theory and Psychological Reality*. The MIT Press.
- Carey, Susan and Elsa Bartlett 1978. Acquiring a single new word. *Papers and reports on Child Language Development*, 15:17–29.
- Clark, Eve 1990. On the pragmatics of contrast. *Journal of Child Language*, 17:417–431.
- Diesendruck, Gil and Lori Markson 2001. Children’s avoidance of lexical overlap: A pragmatic account. *Developmental Psychology*, 37(5):630–641.
- Fazly, Afsaneh, Afra Alishahi, and Suzanne Stevenson 2008. A probabilistic incremental model of word learning in the presence of referential uncertainty. In *Proceedings of the 30th Annual Conference of the Cognitive Science Society*.
- Frank, Michael C., Noah D. Goodman, and Joshua B. Tenenbaum 2007. A bayesian framework for cross-situational word-learning. In *Advances in Neural Information Processing Systems*, volume 20.
- Gershkoff-Stowe, Lisa and Erin R. Hahn 2007. Fast mapping skills in the developing lexicon. *Journal of Speech, Language, and Hearing Research*, 50:682–697.
- Golinkoff, Roberta Michnick, Kathy Hirsh-Pasek, Leslie M. Bailey, and Neil R. Wegner 1992. Young children and adults use lexical principles to learn new nouns. *Developmental Psychology*, 28(1):99–108.
- Gopnik, Alison and Andrew Meltzoff 1987. The development of categorization in the second year and its relation to other cognitive and linguistic developments. *Child Development*, 58(6):1523–1531.
- Halberda, Justin 2006. Is this a dax which I see before me? use of the logical argument disjunctive syllogism supports word-learning in children and adults. *Cognitive Psychology*, 53:310–344.
- Halberda, Justin and Julie Goldman 2008. One-trial learning in 2-year-olds: Children learn new nouns in 3 seconds flat. (in submission).
- Horst, Jessica S., Bob McMurray, and Larissa K. Samuelson 2006. Online processing is essential for learning: Understanding fast mapping and word learning in a dynamic connectionist architecture. In *Proc. of CogSci’06*.
- Horst, Jessica S. and Larissa K. Samuelson 2008. Fast mapping but poor retention by 24-month-old infants. *Infancy*, 13(2):128–157.
- MacWhinney, B. 2000. *The CHILDES Project: Tools for Analyzing Talk*, volume 2: The Database. MahWah, NJ: Lawrence Erlbaum Associates, third edition.
- Markman, Ellen M. and Gwyn F. Wachtel 1988. Children’s use of mutual exclusivity to constrain the meanings of words. *Cognitive Psychology*, 20:121–157.
- Quine, W.V.O. 1960. *Word and Object*. Cambridge, MA: MIT Press.
- Regier, Terry 2005. The emergence of words: Attentional learning in form and meaning. *Cognitive Science*, 29:819–865.
- Reznick, J. Steven and Beverly A. Goldfield 1992. Rapid change in lexical development in comprehension and production. *Developmental Psychology*, 28(3):406–413.
- Siskind, Jeffery Mark 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61:39–91.
- Theakston, A. L., E. V. Lieven, J. M. Pine, and C. F. Rowland 2001. The role of performance limitations in the acquisition of verb-argument structure: An alternative account. *Journal of Child Language*, 28:127–152.

⁷In fact, similar fast mapping effects have been studied in contexts other than language. For example, Behrend et al. (2001) report on children’s fast mapping of novel facts about novel objects.

⁸After processing 1000 input pairs, the average meaning probability of familiar words (those with frequency higher than 10) is 0.77, whereas that of the novel word after one exposure is 0.64.

Improving Word Segmentation by Simultaneously Learning Phonotactics

Daniel Blanchard

Computer & Information Sciences
University of Delaware
dsblanch@udel.edu

Jeffrey Heinz

Linguistics & Cognitive Science
University of Delaware
heinz@udel.edu

Abstract

The most accurate unsupervised word segmentation systems that are currently available (Brent, 1999; Venkataraman, 2001; Goldwater, 2007) use a simple unigram model of phonotactics. While this simplifies some of the calculations, it overlooks cues that infant language acquisition researchers have shown to be useful for segmentation (Mattys et al., 1999; Mattys and Jusczyk, 2001). Here we explore the utility of using bigram and trigram phonotactic models by enhancing Brent’s (1999) MBDP-1 algorithm. The results show the improved MBDP-Phon model outperforms other unsupervised word segmentation systems (e.g., Brent, 1999; Venkataraman, 2001; Goldwater, 2007).

1 Introduction

How do infants come to identify words in the speech stream? As adults, we break up speech into words with such ease that we often think that there are audible pauses between words in the same sentence. However, unlike some written languages, speech does not have any completely reliable markers for the breaks between words (Cole and Jakimik, 1980). In fact, languages vary on how they signal the ends of words (Cutler and Carter, 1987), which makes the task even more daunting. Adults at least have a lexicon they can use to recognize familiar words, but when an infant is first born, they do not have a pre-existing lexicon to consult. In spite of these challenges, by the age of

six months infants can begin to segment words out of speech (Bortfeld et al., 2005). Here we present an efficient word segmentation system aimed to model how infants accomplish the task.

While an algorithm that could reliably extract orthographic representations of both novel and familiar words from acoustic data is something we would like to see developed, following earlier researchers, we simplify the problem by using a text that does not contain any word boundary markers. Hereafter, we use the phrase “word segmentation” to mean some process which adds word boundaries to a text that does not contain them.

This paper’s focus is on unsupervised, incremental word segmentation algorithms; i.e., those that do not rely on preexisting knowledge of a particular language, and those that segment the corpus one utterance at a time. This is in contrast to supervised word segmentation algorithms (e.g., Teahan et al., 2000), which are typically used for segmenting text in documents written in languages that do not put spaces between their words like Chinese. (Of course, unsupervised word segmentation algorithms also have this application.) This also differs from batch segmentation algorithms (Goldwater, 2007; Johnson, 2008b; Fleck, 2008), which process the entire corpus at least once before outputting a segmentation of the corpus. Unsupervised incremental algorithms are of interest to some psycholinguists and acquisitionists interested in the problem of language learning, as well as theoretical computer scientists who are interested in what unsupervised, incremental models are capable of achieving.

Phonotactic patterns are the rules that determine what sequences of phonemes or allophones are allowable within words. Learning the phonotactic patterns of a language is usually modeled

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

separately from word segmentation; e.g., current phonotactic learners such as Coleman and Pierrehumbert (1997), Heinz (2007), or Hayes and Wilson (2008) are given word-sized units as input.

However, infants appear to simultaneously learn which phoneme combinations are allowable within words and how to extract words from the input. It is reasonable that the two processes feed into one another, and when infants acquire a critical mass of phonotactic knowledge, they use it to make judgments about what phoneme sequences can occur within versus across word boundaries (Mattys and Jusczyk, 2001). We use this insight, also suggested by Venkataraman (2001) and recently utilized by Fleck (2008) in a different manner, to enhance Brent’s (1999) model MBDP-1, and significantly increase segmentation accuracy. We call this modified segmentation model MBDP-Phon.

2 Related Work

2.1 Word Segmentation

The problem of unsupervised word segmentation has attracted many earlier researchers over the past fifty years (e.g., Harris, 1954; Olivier, 1968; de Marcken, 1995; Brent, 1999). In this section, we describe the base model MBDP-1, along with two other segmentation approaches, Venkataraman (2001) and Goldwater (2007). In §4, we compare MBDP-Phon to these models in more detail. For a thorough review of word segmentation literature, see Brent (1999) or Goldwater (2007).

2.1.1 MBDP-1

Brent’s (1999) MBDP-1 (Model Based Dynamic Programming) algorithm is an implementation of the INCDROP framework (Brent, 1997) that uses a Bayesian model of how to generate an unsegmented text to insert word boundaries. The generative model consists of five steps:

1. Choose a number of word types, n .
2. Pick n distinct strings from $\Sigma^+ \#$, which will make up the lexicon, L . Entries in L are labeled $W_1 \dots W_n$. $W_0 = \$$, where $\$$ is the utterance boundary marker.
3. Pick a function, f , which maps word types to their frequency in the text.
4. Choose a function, s , to map positions in the text to word types.

5. Concatenate the words in the order specified by s , and remove the word delimiters ($\#$).

It is important to note that this model treats the generation of the text as a single event in the probability space, which allows Brent to make a number of simplifying assumptions. As the values for n , L , f , and s completely determine the segmentation, the probability of a particular segmentation, \bar{w}_m , can be calculated as:

$$P(\bar{w}_m) = P(n, L, f, s) \quad (1)$$

To allow the model to operate on one utterance at a time, Brent states the probability of each word in the text as a recursive function, $R(\bar{w}_k)$, where \bar{w}_k is the text up to and including the word at position k , w_k . Furthermore, there are two specific cases for R : familiar words and novel words. If w_k is familiar, the model already has the word in its lexicon, and its score is calculated as in Equation 2.

$$R(\bar{w}_k) = \frac{f(w_k)}{k} \cdot \left(\frac{f(w_k) - 1}{f(w_k)} \right)^2 \quad (2)$$

Otherwise, the word is novel, and its score is calculated using Equation 3¹ (Brent and Tao, 2001),

$$R(\bar{w}_k) = \frac{6}{\pi^2} \cdot \frac{n}{k} \cdot \frac{P_\Sigma(a_1) \dots P_\Sigma(a_q)}{1 - P_\Sigma(\#)} \cdot \left(\frac{n-1}{n} \right)^2 \quad (3)$$

where P_Σ is the probability of a particular phoneme occurring in the text. The third term of the equation for novel words is where the model’s unigram phonotactic model comes into play. We detail how to plug a more sophisticated phonotactic learning model into this equation in §3. With the generative model established, MBDP-1 uses a Viterbi-style search algorithm to find the segmentation for each utterance that maximizes the R values for each word in the segmentation.

Venkataraman (2001) notes that considering the generation of the text as a single event is unlikely to be how infants approach the segmentation problem. However, MBDP-1 uses an incremental search algorithm to segment one utterance at a time, which is more plausible as a model of infants’ word segmentation.

¹Brent (1999) originally described the novel word score as $R(\bar{w}_k) = \frac{6}{\pi^2} \cdot \frac{n_k}{k} \cdot \frac{P_\sigma(W_{n_k})}{1 - \frac{n_k-1}{n_k} \cdot \sum_{j=1}^{n_k} P_\sigma(W_j)} \cdot \left(\frac{n_k-1}{n_k} \right)^2$, where P_σ is the probability of all the phonemes in the word occurring together, but the denominator of the third term was dropped in Brent and Tao (2001). This change drastically speeds up the model, and only reduces segmentation accuracy by $\sim 0.5\%$.

2.1.2 Venkataraman (2001)

MBDP-1 is not the only incremental unsupervised segmentation model that achieves promising results. Venkataraman’s (2001) model tracks MBDP-1’s performance so closely that Batchelder (2002) posits that the models are performing the same operations, even though the authors describe them differently.

Venkataraman’s model uses a more traditional, smoothed n-gram model to describe the distribution of words in an unsegmented text.² The most probable segmentation is retrieved via a dynamic programming algorithm, much like Brent (1999).

We use MBDP-1 rather than Venkataraman’s approach as the basis for our model only because it was more transparent how to plug in a phonotactic learning module at the time this project began.

2.1.3 Goldwater (2007)

We also compare our results to a segmenter put forward by Goldwater (2007). Goldwater’s segmenter uses an underlying generative model, much like MBDP-1 does, only her language model is described as a Dirichlet process (see also Johnson, 2008b). While this model uses a unigram model of phoneme distribution, as did MBDP-1, it implements a bigram word model like Venkataraman (2001). A bigram word model is useful in that it prevents the segmenter from assuming that frequent word bigrams are not simply one word, which Goldwater observes happen with a unigram version of her model.

Goldwater uses a Gibbs sampler augmented with simulated annealing to sample from the posterior distribution of segmentations and determine the most likely segmentation of each utterance.³ This approach requires non-incremental learning.⁴ We include comparison with Goldwater’s segmenter because it outperforms MBDP-1 and Venkataraman (2001) in both precision and recall, and we are interested in whether an incremental algorithm supplemented with phonotactic learning can match its performance.

2.2 Phonotactic Learning

Phonotactic acquisition models have seen a surge in popularity recently (e.g., Coleman and Pierre-

²We refer the reader to Venkataraman (2001) for the details of this approach.

³We direct the reader to Goldwater (2007) for details.

⁴In our experiments and those in Goldwater (2007), the segmenter runs through the corpus 1000 times before outputting the final segmentation.

humbert, 1997; Heinz, 2007; Hayes and Wilson, 2008). While Hayes and Wilson present a more complex Maximum Entropy phonotactic model in their paper than the one we add to MBDP-1, they also evaluate a simple n-gram phonotactic learner operating over phonemes. The input to the models is a list of English onsets and their frequency in the lexicon, and the basic trigram learner simply keeps track of the trigrams it has seen in the corpus. They test the model on novel words with acceptable rhymes—some well-formed (e.g., [kɪp]), and some less well-formed (e.g., [stwi:k])—so any ill-formedness is attributable to onsets. This basic trigram model explains 87.7% of the variance in the scores that Scholes (1966) reports his 7th grade students gave when subjected to the same test. When Hayes and Wilson run their Maximum Entropy phonotactic learning model with n-grams over phonological features, the r-score increases substantially to 95.6%.

Given the success and simplicity of the basic n-gram phonotactic model, we choose to integrate this with MBDP-1.

3 Extending MBDP-1 with Phonotactics

The main contribution of our work is adding a phonotactic learning component to MBDP-1 (Brent, 1999). As we mention in §2.1.1, the third term of Equation 3 is where MBDP-1’s unigram phonotactic assumption surfaces. The original model simply multiplies the probabilities of all the phonemes in the word together and divides by one minus the probability of a particular phoneme being the word boundary to come up with probability of the phoneme combination. The order of the phonemes in the word has no effect on its score. The only change we make to MBDP-1 is to the third term of Equation 3. In MBDP-Phon this becomes

$$\prod_{i=0}^q P_{\text{MLE}}(a_i \dots a_j) \quad (4)$$

where $a_i \dots a_j$ is an n-gram inside a proposed word, and a_0 and a_q are both the word boundary symbol, #⁵.

It is important to note that probabilities calculated in Equation 4 are maximum likelihood estimates of the joint probability of each n-gram in the word. The maximum likelihood estimate (MLE)

⁵The model treats word boundary markers like a phoneme for the purposes of storing n-grams (i.e., a word boundary marker may occur anywhere within the n-grams).

for a particular n-gram inside a word is calculated by dividing the total number of occurrences of that n-gram (including in the word we are currently examining) by the total number of n-grams (including those in the current word). The numbers of n-grams are computed with respect to the obtained lexicon, not the corpus, and thus the frequency of lexical items in the corpus does not affect the n-gram counts, just like Brent’s unigram phonotactic model and other phonotactic learning models (e.g., Hayes and Wilson, 2008).

We use the joint probability instead of the conditional probability which is often used in computational linguistics (Manning and Schütze, 1999; Jurafsky and Martin, 2000), because of our intuition that the joint probability is truer to the idea that a phonotactically well-formed word is made up of n-grams that occur frequently in the lexicon. On the other hand, the conditional probability is used when one tries to predict the next phoneme that will occur in a word, rather than judging the well-formedness of the word as a whole.⁶

We are able to drop the denominator that was originally in Equation 3, because $P_{\Sigma}(\#)$ is zero for an n -gram model when $n > 1$. This simple modification allows the model to learn what phonemes are more likely to occur at the beginnings and ends of words, and what combinations of phonemes rarely occur within words.

What is especially interesting about this modification is that the phonotactic learning component estimates the probabilities of the n-grams by using their relative frequencies in the words the segmenter has extracted. The phonotactic learner is guaranteed to see at least two valid patterns in every utterance, as the n-grams that occur at the beginnings and ends of utterances are definitely at the beginnings and ends of words. This allows the learner to provide useful information to the segmenter even early on, and as the segmenter correctly identifies more words, the phonotactic learner has more correct data to learn from. Not only is this mutually beneficial process supported by evidence from language acquisitionists (Mattys et al., 1999; Mattys and Jusczyk, 2001), it also resembles co-training (Blum and Mitchell, 1998). We refer to the extended version of Brent’s model

⁶This intuition is backed up by preliminary results suggesting MBDP-Phon performs better when using MLEs of the joint probability as opposed to conditional probability. There is an interesting question here, which is beyond the scope of this paper, so we leave it for future investigation.

described above as MBDP-Phon.

4 Evaluation

4.1 The Corpus

We run all of our experiments on the Bernstein-Ratner (1987) infant-directed speech corpus from the CHILDES database (MacWhinney and Snow, 1985). This is the same corpus that Brent (1999), Goldwater (2007), and Venkataraman (2001) evaluate their models on, and it has become the *de facto* standard for segmentation testing, as unlike other corpora in CHILDES, it was phonetically transcribed.

We examine the transcription system Brent (1999) uses and conclude some unorthodox choices were made when transcribing the corpus. Specifically, some phonemes that are normally considered distinct are combined into one symbol, which we call a bi-phone symbol. These phonemes combinations include diphthongs and vowels followed by /ɪ/. Another seemingly arbitrary decision is the distinction between stressed and unstressed syllabic /ɪ/ sound (i.e., there are different symbols for the /ɪ/ in “butter” and the /ɪ/ in “bird”) since stress is not marked elsewhere in the corpus. To see the effect of these decisions, we modified the corpus so that the bi-phone symbols were split into two⁷ and the syllabic /ɪ/ symbols were collapsed into one.

4.2 Accuracy

We ran MBDP-1 on the original corpus, and the modified version of the corpus. As illustrated by Figures 1 and 2, MBDP-1 performs worse on the modified corpus with respect to both precision and recall. As MBDP-1 and MBDP-Phon are both iterative learners, we calculate segmentation precision and recall values over 500-utterance blocks. Per Brent (1999) and Goldwater (2007), precision and recall scores reflect correctly segmented words, not correctly identified boundaries.

We also test to see how the addition of an n-gram phonotactic model affects the segmentation accuracy of MBDP-Phon by comparing it to MBDP-1 on our modified corpus.⁸ As seen in Figure 3, MBDP-Phon using bigrams (henceforth MBDP-Phon-Bigrams) is consistently more precise in its

⁷We only split diphthongs whose first phoneme can occur in isolation in English, so the vowels in “bay” and “boat” were not split.

⁸We also compare MBDP-Phon to MBDP-1 on the original corpus. The results are given in Tables 1 and 2.

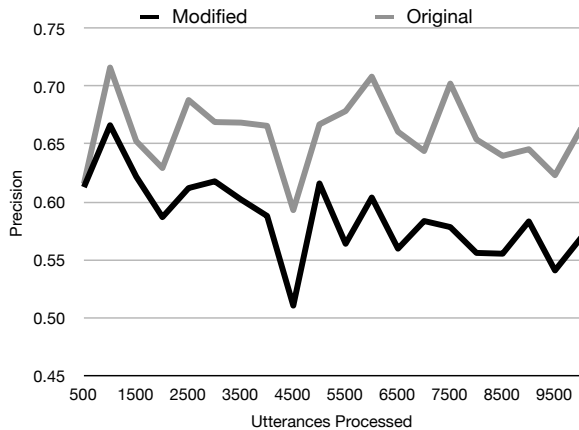


Figure 1: Precision of MBDP-1 on both corpora.

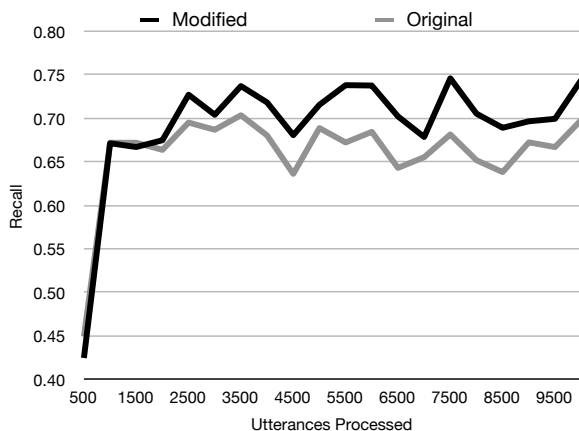


Figure 2: Recall of MBDP-1 on both corpora.

segmentation than MBDP-1, and bests it by $\sim 18\%$ in the last block. Furthermore, MBDP-Phon-Bigrams significantly outpaces MBDP-1 with respect to recall only after seeing 1000 utterances, and finishes the corpus $\sim 10\%$ ahead of MBDP-1 (see Figure 4). MBDP-Phon-Trigrams does not fair as well in our tests, falling behind MBDP-1 and MBDP-Phon-Bigrams in recall, and MBDP-Phon-Bigrams in precision. We attribute this poor performance to the fact that we are not currently smoothing the n-gram models in any way, which leads to data sparsity issues when using trigrams. We discuss a potential solution to this problem in §5.

Having established that MBDP-Phon-Bigrams significantly outperforms MBDP-1, we compare its segmentation accuracy to those of Goldwater (2007) and Venkataraman (2001).⁹ As before, we

⁹We only examine Venkataraman’s unigram model, as his bigram and trigram models perform better on precision, but worse on recall.

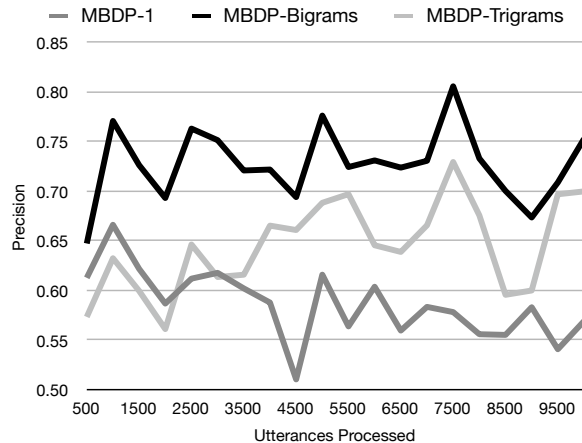


Figure 3: Precision of MBDP-1 and MBDP-Phon on modified corpus.

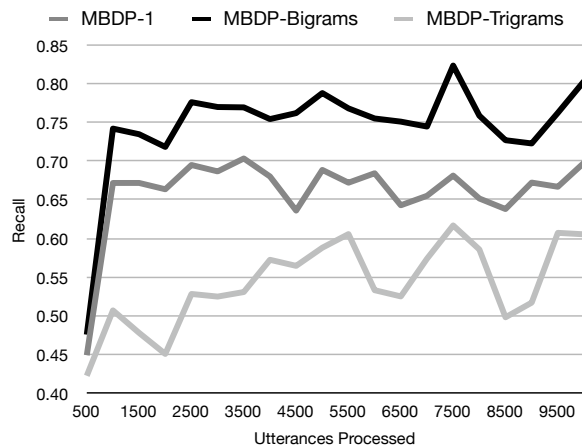


Figure 4: Recall of MBDP-1 and MBDP-Phon on modified corpus.

run the models on the entire corpus, and then measure their performance over 500-utterance blocks.

MBDP-Phon-Bigrams edges out Goldwater’s model in precision on our modified corpus, with an average precision of 72.79% vs. Goldwater’s 70.73% (Table 1). If we drop the first 500-utterance block for MBDP-Phon-Bigrams because the model is still in the early learning stages, whereas Goldwater’s has seen the entire corpus, its average precision increases to 73.21% (Table 1). When considering the recall scores in Table 2, it becomes clear that MBDP-Phon-Bigrams has a clear advantage over the other models. Its average recall is higher than or nearly equal to both of the other models’ maximum scores. Since Venkataraman’s (2001) model performs similarly to MBDP-1, it is no surprise that MBDP-Phon-Bigrams achieves higher precision and recall.

	MBDP-Phon-Bigrams	Venkataraman	Goldwater
	<i>Original: Utterances 0 to 9790</i>		
Avg.	72.84%	67.46%	67.87%
Max.	79.91%	71.79%	71.98%
Min.	63.97%	61.77%	61.87%
	<i>Modified: Utterances 0 to 9790</i>		
Avg.	72.79%	59.64%	70.73%
Max.	80.60%	66.84%	74.61%
Min.	64.78%	52.54%	65.29%
	<i>Modified: Utterances 500 to 9790</i>		
Avg.	73.21%	59.54%	70.59%
Max.	80.60%	66.84%	74.61%
Min.	67.40%	52.54%	65.29%

Table 1: Precision statistics for MBDP-Phon-Bigrams, Goldwater, and Venkataraman on both corpora over 500-utterance blocks.

The only metric by which MBDP-Phon-Bigrams does not outperform the other algorithms is lexical precision, as shown in Table 3. Lexical precision is the ratio of the number of correctly identified words in the lexicon to the total number of words in the lexicon (Brent, 1999; Venkataraman, 2001).¹⁰ The relatively poor performance of MBDP-Phon-Bigrams is due to the incremental nature of the MBDP algorithm. Initially, it makes numerous incorrect guesses that are added to the lexicon, and there is no point at which the lexicon is purged of earlier erroneous guesses (c.f. the improved lexical precision when omitting the first block in Table 3). On the other hand, Goldwater’s algorithm runs over the corpus multiple times, and only produces output when it settles on a final segmentation.

In sum, MBDP-Phon-Bigrams significantly improves the accuracy of MBDP-1, and achieves better performance than the models described in Venkataraman (2001) and Goldwater (2007).

5 Future Work

There are many ways to implement phonotactic learning. One idea is to use n-grams over phonological features, as per Hayes and Wilson (2008). Preliminary results have shown that we need to add smoothing to our n-gram model, and we plan to use

¹⁰See Brent (1999) for a discussion of the meaning of this statistic.

	MBDP-Phon-Bigrams	Venkataraman	Goldwater
	<i>Original: Utterances 0 to 9790</i>		
Avg.	72.03%	70.02%	71.02%
Max.	79.31%	75.59%	76.79%
Min.	44.71%	42.57%	64.32%
	<i>Modified: Utterances 0 to 9790</i>		
Avg.	74.63%	66.24%	70.48%
Max.	82.45%	70.47%	74.79%
Min.	47.63%	44.71%	63.74%
	<i>Modified: Utterances 500 to 9790</i>		
Avg.	76.05%	67.37%	70.28%
Max.	82.45%	70.47%	74.79%
Min.	71.92%	63.86%	63.74%

Table 2: Recall statistics for MBDP-Phon-Bigrams, Goldwater, and Venkataraman on both corpora over 500-utterance blocks.

Modified Kneser-Ney smoothing (Chen and Goodman, 1998).

Another approach would be to develop a syllable-based phonotactic model (Coleman and Pierrehumbert, 1997). Johnson (2008b) achieves impressive segmentation results by adding a syllable level with Adaptor grammars.

Some languages (e.g., Finnish, and Navajo) contain long-distance phonotactic constraints that cannot be learned by n-gram learners (Heinz, 2007). Heinz (2007) shows that precedence-based learners—which work like a bigram model, but without the restriction that the elements in the bigram be adjacent—can handle many long-distance agreement patterns (e.g., vowel and consonantal harmony) in the world’s languages. We posit that adding such a learner to MBDP-Phon would allow it to handle a greater variety of languages.

Since none of these approaches to phonotactic learning depend on MBDP-1, it is also of interest to integrate phonotactic learners with other word segmentation strategies.

In addition to evaluating segmentation models integrated with phonotactic learning on their segmentation performance, it would be interesting to evaluate the quality of the phonotactic grammars obtained. A good point of comparison for English are the constraints obtained by Hayes and Wilson (2008), since the data with which they tested their phonotactic learner is publicly available.

Finally, we are looking forward to investigat-

	MBDP-Phon-Bigrams	Venkataraman	Goldwater
<i>Original: Utterances 0 to 9790</i>			
Avg.	47.69%	49.78%	56.50%
Max.	49.71%	52.95%	63.09%
Min.	46.30%	41.83%	55.33%
<i>Modified: Utterances 0 to 9790</i>			
Avg.	48.31%	45.98%	58.03%
Max.	50.42%	48.90%	65.58%
Min.	41.74%	36.57%	56.43%
<i>Modified: Utterances 500 to 9790</i>			
Avg.	54.34%	53.06%	57.95%
Max.	63.76%	54.35%	62.30%
Min.	51.31%	51.95%	56.52%

Table 3: Lexical precision statistics for MBDP-Phon-Bigrams, Goldwater, and Venkataraman on both corpora over 500-utterance blocks.

ing the abilities of these segmenters on corpora of different languages. Fleck (2008) tests her segmenter on a number of corpora, including Arabic and Spanish, and Johnson (2008a) applies his segmenter to a corpus of Sesotho.

6 Conclusion

From the results established in §4, we can conclude that MBDP-Phon using a bigram phonotactic model is more accurate than the models described in Brent (1999), Venkataraman (2001), and Goldwater (2007). The n-gram phonotactic model improves overall performance, and is especially useful for corpora that do not encode diphthongs with bi-phone symbols. The main reason there is such a marked improvement with MBDP-Phon vs. MBDP-1 when the bi-phone symbols were removed from the original corpus is that these bi-phone symbols effectively allow MBDP-1 to have a select few bigrams in the cases where it would otherwise over-segment.

The success of MBDP-Phon is not clear evidence that the INCDROP framework (Brent, 1997) is superior to Venkataraman or Goldwater’s models. We imagine that adding a phonotactic learning component to either of their models would also improve their performance.

We also tentatively conclude that phonotactic patterns can be learned from unsegmented text. However, the phonotactic patterns learned by our model ought to be studied in detail to see how well

they match the phonotactic patterns of English.

MBDP-Phon’s performance reinforces the theory put forward by language acquisition researchers that phonotactic knowledge is a cue for word segmentation (Mattys et al., 1999; Mattys and Jusczyk, 2001). Furthermore, our results indicate that learning phonotactic patterns can occur simultaneously with word segmentation. Finally, further investigation of the simultaneous acquisition of phonotactics and word segmentation appears fruitful for theoretical and computational linguists, as well as acquisitionists.

Acknowledgements

We are grateful to Roberta Golinkoff who inspired this project. We also thank Vijay Shanker for valuable discussion, Michael Brent for the corpus, and Sharon Goldwater for the latest version of her code.

References

- Batchelder, Eleanor Olds. 2002. Bootstrapping the lexicon: a computational model of infant speech segmentation. *Cognition*, 83(2):167–206.
- Bernstein-Ratner, Nan. 1987. *The phonology of parent child speech*, volume 6. Erlbaum, Hillsdale, NJ.
- Blum, Avrim and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Workshop on Computational Learning Theory*, pages 92–100.
- Bortfeld, Heather, James Morgan, Roberta Golinkoff, and Karen Rathbun. 2005. Mommy and me: Familiar names help launch babies into speech-stream segmentation. *Psychological Science*, 16(4):298–304.
- Brent, Michael R. 1997. Towards a unified model of lexical acquisition and lexical access. *Journal of Psycholinguistic Research*, 26(3):363–375.
- Brent, Michael R. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.
- Brent, Michael R and Xiaopeng Tao. 2001. Chinese text segmentation with mbdp-1: Making the most of training corpora. In *39th Annual Meeting of the ACL*, pages 82–89.
- Chen, Stanley F and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98,

- Center for Research in Computing Technology, Harvard University.
- Cole, Ronald and Jola Jakimik. 1980. *A model of speech perception*, pages 136–163. Lawrence Erlbaum Associates, Hillsdale, NJ.
- Coleman, John and Janet Pierrehumbert. 1997. Stochastic phonological grammars and acceptability. In *Third Meeting of the ACL SIGPHON*, pages 49–56. ACL, Somerset, NJ.
- Cutler, Anne and David Carter. 1987. The predominance of strong initial syllables in the english vocabulary. *Computer Speech and Language*, 2(3-4):133–142.
- de Marcken, Carl. 1995. Acquiring a lexicon from unsegmented speech. In *33rd Annual Meeting of the ACL*, pages 311–313.
- Fleck, Margaret M. 2008. Lexicalized phonotactic word segmentation. In *46th Annual Meeting of the ACL*, pages 130–138. ACL, Morristown, NJ.
- Goldwater, Sharon. 2007. *Nonparametric Bayesian Models of Lexical Acquisition*. Ph.D. thesis, Brown University, Department of Cognitive and Linguistic Sciences.
- Harris, Zellig. 1954. Distributional structure. *Word*, 10(2/3):146–62.
- Hayes, Bruce and Colin Wilson. 2008. A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*.
- Heinz, Jeffrey. 2007. *Inductive Learning of Phonotactic Patterns*. Ph.D. thesis, University of California, Los Angeles, Department of Linguistics.
- Johnson, Mark. 2008a. Unsupervised word segmentation for sesotho using adaptor grammars. In *Tenth Meeting of ACL SIGMORPHON*, pages 20–27. ACL, Morristown, NJ.
- Johnson, Mark. 2008b. Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *46th Annual Meeting of the ACL*, pages 398–406. ACL, Morristown, NJ.
- Jurafsky, Daniel and James Martin. 2000. *Speech and Language Processing*. Prentice-Hall.
- MacWhinney, Brian and Catherine Snow. 1985. The child language data exchange system. *Journal of child language*, 12(2):271–95.
- Manning, Christopher and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.
- Mattys, Sven and Peter Jusczyk. 2001. Phonotactic cues for segmentation of fluent speech by infants. *Cognition*, 78:91–121.
- Mattys, Sven, Peter Jusczyk, Paul Luce, and James Morgan. 1999. Phonotactic and prosodic effects on word segmentation in infants. *Cognitive Psychology*, 38:465–494.
- Olivier, Donald. 1968. *Stochastic Grammars and Language Acquisition Mechanisms*. Ph.D. thesis, Harvard University.
- Scholes, Robert. 1966. *Phonotactic Grammaticality*. Mouton, The Hague.
- Teahan, W. J., Rodger McNab, Yingying Wen, and Ian H. Witten. 2000. A compression-based algorithm for chinese word segmentation. *Computational Linguistics*, 26(3):375–393.
- Venkataraman, Anand. 2001. A statistical model for word discovery in transcribed speech. *Computational Linguistics*, 27(3):352–372.

A MDL-based Model of Gender Knowledge Acquisition

Harmony Marchal¹, Benoît Lemaire², Maryse Bianco¹, and Philippe Dessus¹

¹L.S.E. and ²Laboratoire TIMC-IMAG

University of Grenoble, FRANCE

<first name>.<last name>@upmf-grenoble.fr

Abstract

This paper presents an iterative model of knowledge acquisition of gender information associated with word endings in French. Gender knowledge is represented as a set of rules containing exceptions. Our model takes noun-gender pairs as input and constantly maintains a list of rules and exceptions which is both coherent with the input data and minimal with respect to a minimum description length criterion. This model was compared to human data at various ages and showed a good fit. We also compared the kind of rules discovered by the model with rules usually extracted by linguists and found interesting discrepancies.

1 Introduction

In several languages, nouns have a gender. In French, nouns are either masculine or feminine. For example, you should say *le camion* (the truck) but *la voiture* (the car). Gender assignment in French can be performed using two kinds of information. Firstly, *lexical information*, related to the co-occurring words (e.g., articles, adjectives) which most of times marks gender unambiguously. Secondly, *sublexical information*, especially noun-endings, are pretty good predictors of their grammatical gender (e.g., almost all nouns endings in *-age* are masculine). Several word endings can be used to reliably predict gender of new words but this kind of rules is never explicitly taught to children: they have to implicitly learn that knowledge from exposure to noun-gender pairs. It turns out that children as young as 3 already constructed some of these

rules, which can be observed by testing them on pseudo-words (Karmiloff-Smith, 1979).

This paper presents an iterative model of the way children may acquire this gender knowledge. Its input is a large random sequence of noun-gender pairs following the distribution of word frequency at a given age. It is supposed to represent the words children are exposed to. The model constantly maintains a list of rules and exceptions both coherent with the input data and minimal with respect to an information theory criterion. This model was compared to human data at various ages and showed a good fit. We also compared the kind of rules discovered by the model with rules usually extracted by linguists and found interesting discrepancies.

2 Principle of Simplicity

Gender knowledge is learned from examples. Children are exposed to thousands of nouns which are most of the time accompanied with a gender clue because of their corresponding determiner or adjective. For instance, when hearing “*ta poussette est derrière le fauteuil*” [your stroller is behind the armchair], a child knows that *poussette* is feminine because of the feminine possessive determiner *ta*, and that *fauteuil* is masculine because of the masculine determiner *le*. After processing thousands of such noun/gender pairs, children acquired some gender knowledge which allows them to predict the gender of pseudo-words (Marchal et al., 2007; Meunier et al., 2008). This knowledge is largely dependent on the end of the words since the endings of many nouns in French are associated more often with one gender than the other (Holmes & Segui, 2004). For instance children would predict that pseudo-words such as *limette* or *mossette* are rather feminine words although they never heard them before. It means that they should have constructed a rule-like knowledge saying that “*words ending in -ette are rather feminine*”. Or maybe it is “*words ending in -te are rather feminine*” or even “*words ending in -e*

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

are rather feminine”... Actually, there are many ways to structure this knowledge, especially because this kind of rule generally has exceptions. Let us take an example. Consider the following words and their gender (masculine or feminine): *barrage* [weir] (m), *image* [image] (f), *courage* [courage] (m), *plage* [beach] (f), *étage* [floor] (m), *garage* [garage] (m), *collage* [collage] (m). Several rules could be constructed from this data:

- (1) words ending in *-age* are masculine except *image* and *plage*;
- (2) words ending in *-age* are feminine except *barrage*, *courage*, *étage*, *garage* and *collage*;
- (3) words ending in *-age* are feminine except words ending in *-rage*, *étage* and *collage*.

The latter is an example of a rule whose exceptions may themselves contain rules. The question is to know which rules may be constructed and used by children, and which cognitive mechanisms may lead to the construction of such rules. In order to investigate that issue, we relied on the assumption that children minds obey a principle of simplicity.

This principle is a cognitive implementation of the Occam’s razor, saying that one should choose the simplest hypothesis consistent with the data. This idea has already been used in the field of concept learning where it would dictate that we induce the simplest category consistent with the observed examples—the most parsimonious generalization available (Feldman, 2003). Chater & Vitányi (2003) view it as a unifying principle in cognitive science to solve the problem of induction in which infinitely many patterns are compatible with any finite set of data. They assume “that the learner chooses the underlying theory of the probabilistic structure of the language that provides the simplest explanation of the history of linguistic input to which the learner has been exposed.” (Chater & Vitányi, 2007).

One way to implement this idea is to consider that the simplest description of a hypothesis is the shortest one. Without considering frequency of the rule usage, rule 1 in the previous example seems intuitively more likely to be used by humans because it is the shortest.

Intuitively, counting the number of characters of each hypothesis could seem a good method but it is better to choose the most compact representation (Chater, 1999). More important, the choice should also depend on the frequency of rule usage: the description length of a rule that would be frequently used should not be counted

like a seldom used rule. For instance, rule 2 could be a more appropriate coding if it is used very frequently in the language as opposed to the frequency of its exceptions. That is the reason why we rely on word frequencies for various ages in our simulations.

Information theory provides a formal version of this assumption: the minimum description length (MDL) principle (Rissanen, 1978). The goal is to minimize the coding cost of both the hypothesis and the data reconstructed from the hypothesis (two-part coding). However, we will see that, in our case, the model contains all the data which lead to a simpler mechanism: the idea is to select the hypothesis which represents the data in the most compact way, that is which has the shortest code length. Given a realization x of a random variable X with probability distribution p , x can be optimally coded with a size of $-\log_2(p(x))$ bits.

For instance, suppose you are exposed to only 4 words A, B, C and D with frequencies .5, .25, .125, .125. For example, exposure could be: BAACADBABACADBAA. An optimal coding would need only 1 bit ($-\log_2(.5)$) to code word A since it occurs 50% of the time. For instance, A would be **0** and all other words would begin with **1**. B needs 2 bits ($-\log_2(.25)$), for instance **10**. C and D both needs 3 bits ($-\log_2(.125)$), for instance **110** for C and **111** for D.

The average code length for a realization of the random variable X is computed by weighting each code length by the corresponding probability. It is exactly what is called entropy:

$$H(X) = - \sum p(x) \cdot \log_2(p(x))$$

In the previous example, the average code length is $1 \times .5 + 2 \times .25 + 3 \times .125 + 3 \times .125 = 1.75$ bits

From this point of view, learning is data compression (Grünwald, 2005). To sum up, the general idea of our approach is to generate rules that are coherent with the data observed so far and to select the one with the smallest entropy.

3 Model

Some computational models have been proposed in the literature, but they are concerned with the problem of gender assignment given an existing lexicon rather than dynamically modeling the acquisition of gender knowledge. Their input is therefore a set of words representative of all the words in the language. Analogical modeling (Skousen, 2003) is such a model. It predicts the gender of a new word by constructing a set of words that are analogous to it, with respect to

morphology. Matthews (2005) compared analogical modeling and a neural net and could not find any significant difference. Our model takes noun-gender pairs as input and dynamically updates the set of rules it has constructed so far in order to minimize their description length.

3.1 Input

The input to our model is supposed to represent the noun/gender pairs children are exposed to. We used *Manulex* (Lété et al., 2004), a French lexical database which contains word frequencies of 48,900 lexical forms from the analysis of 54 textbooks. Word frequencies are provided for 3 levels: grades 1, 2 and 3-5.

We used the phonetic form of words² because the development of the gender knowledge is only based on phonological data during the first six years of life. It would also be interesting to study the development of written-specific rules, but this will be done in a future work.

We constructed a learning corpus by randomly selecting in this database 200,000 words and their gender such that their distribution is akin to their frequency distribution in *Manulex*. In other words, the probability of picking a given word in the corpus is just its frequency. In fact, we suppose that the construction of the rule depends on the frequency of words children are exposed to and not just on the words at a type level.

It would have been more accurate to take real corpora as input, in particular because the order in which words are considered probably plays a role, but such French corpora for specific ages, large enough to be sufficiently accurate, do not exist to our knowledge.

We now present how our model handles these noun-gender pairs, one after the other.

3.2 Knowledge Representation

Gender knowledge is represented as rules containing exceptions. The premise of a rule is a word ending and the conclusion is a gender. The * character indicates any substring preceding the word ending. A natural language example of a rule is:

- (4) */yR/ are feminine nouns (f) except /azyR/, /myR/, /myRmyR/ which are masculine (m).

² We used an ASCII version of the International Phonetic Alphabet.

Exceptions may contain words that could also be organized in rules, which itself may contain exceptions. Here is an example:

- (5) */R/→m except:
 /tiRliR/, /istwaR/→f
 */jER/→f except /gRyjER/→m
 */yR/→f except /azyR/ and /myR/→m

The gender knowledge corresponding to a given corpus is represented as a set of such rules. Such a set contains about 80 rules for a grade-1 learning corpus. We now present how this knowledge is updated according to a new noun-gender pair to be processed.

3.3 Rule Construction

Each time a new noun-gender pair is processed, all possible set of rules that are coherent with the data are generated, and the best one, with respect to the minimum description length criterion, will be selected. As an example, consider this little current set of two rules which was constructed from the words /azyR/, /baRaZ/, /etaZ/, /imaZ/, /plaZ/, /SosyR/ and /vwAtyR/³ (words above below square brackets are the examples which were used to form the rule):

- (6) */yR/→f [/SosyR/, /vwAtyR/] except /azyR/→m
 (7a) */aZ/→f [/imaZ/, /plaZ/] except /etaZ/, /baRaZ/→m

Then a new word is processed: /kuRaZ/ which is of masculine gender. Since it is not coherent with the most specific rule (rule 7a) matching its ending (genders are different), the algorithm attempts to generalize it with the first-level exceptions in order to make a new rule. /etaZ/ is taken first. It can be generalized with the new word /kuRaZ/ to form the new rule:

- (8a) */aZ/→m [/etaZ/, /kuRaZ/]

All other exceptions which could be included are added. The new rule becomes:

- (8b) */aZ/→m [/baRaZ/, /etaZ/, /kuRaZ/]

Once a new rule has been created, the algorithm needs to maintain the coherence of the base. It checks whether this new rule is in conflict with other rules with a *different gender*. This is the

³ Translations: /azyR/ (*azur* [azure]), /baRaZ/ (*barage* [weir]), /etaZ/ (*étage* [floor]), /imaZ/ (*image* [image]), /plaZ/ (*plage* [beach]), /SosyR/ (*chaussure* [shoe]) and /vwAtyR/ (*voiture* [car])

case since we have the exact same rule but for the feminine gender (rule 7a). Conflicting examples are therefore removed from the old rule and put as exceptions to the new rule. In that case of identity between old and new rule, all examples are removed and the rule disappears. The new rule is:

(8c) $*/aZ/\rightarrow m$ [/baRaZ/, /etaZ/, /kuRaZ/] except /imaZ/, /plaZ/ $\rightarrow f$

After having checked for rules with a *different gender*, the algorithm now checks for existing rules with the *same gender* that the new rule, either more specific or more general. This is not the case here. We thus created our first candidate set of rules (rules 6 and 8c):

CANDIDATE SET #1:

$*/yR/\rightarrow f$ [/SosyR/, /vwAtyR/] except /azyR/ $\rightarrow m$
 $*/aZ/\rightarrow m$ [/baRaZ/, /etaZ/, /kuRaZ/] except /imaZ/, /plaZ/ $\rightarrow f$

Other rules could have been generated from the set of exceptions of $*/aZ/\rightarrow f$. The word /etaZ/ was taken first but the algorithm needs to consider all other exceptions. It then takes /baRaZ/ to form the rule:

(9) $*/RaZ/\rightarrow m$ [/baRaZ/, /kuRaZ/]

Note that this is a more specific rule than the previous one: it is based on a 3-letter ending whereas /etaZ/ and /kuRaZ/ generated a 2-letter ending. No other exceptions can be added. The algorithm now checks for conflicting rules with the same gender and puts this new rule as an exception of the previous rule. Then it checks for possible conflict with rules of different gender, but there are none. The second candidate set is therefore:

CANDIDATE SET #2:

$*/yR/\rightarrow f$ [/SosyR/, /vwAtyR/] except /azyR/ $\rightarrow m$
 $*/aZ/\rightarrow f$ [/imaZ/, /plaZ/] except /etaZ/ $\rightarrow m$
 $*/RaZ/$ [/baRaZ/, /kuRaZ/] $\rightarrow m$

Something else needs to be done: removing words from a rule and putting them as exceptions may lead to new generalizations between them or with other existing words. In our case, the algorithm memorized the fact that /imaZ/ and /plaZ/ have been put as exceptions.

It now applies the same mechanism as before: adding those words to the new set of rules, as if they were new words. By the same previous algorithm, it gives the new rule:

(7b) $*/aZ/\rightarrow f$ [/imaZ/, /plaZ/]

In order to maintain the coherence of the rule base, examples of conflicting rules are removed and put as exceptions:

(7c) $*/aZ/\rightarrow f$ [/imaZ/, /plaZ/] except /baRaZ/, /etaZ/, /kuRaZ/ $\rightarrow m$

We now have our third candidate set of rules:

CANDIDATE SET #3:

$*/yR/\rightarrow f$ [/SosyR/, /vwAtyR/] except /azyR/ $\rightarrow m$
 $*/aZ/\rightarrow f$ [/imaZ,plaZ] except /etaZ/, /baRaZ/, /kuRaZ/ $\rightarrow m$

Figure 1 summarizes the model's architecture.

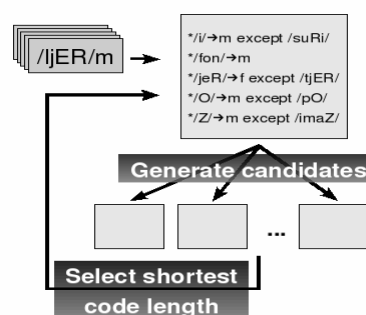


Figure 1. Overall architecture

3.4 Model Selection

This section describes how to choose between candidate models. As we mentioned before, the idea is to select the most compact model. For each exception, we compute its frequency F from the number of times it appeared so far. For each rule, F is just the sum of the frequencies of all examples it covered.

The description length of each rule or exception is $-\log_2(F)$. Since the overall value needs to take into account the variation of frequency of each rule or exception, each description length is weighted by its frequency, which gives the average description length of a candidate set of rules (corresponding to the entropy):

$$weight(Model) = -\sum F_i \cdot \log_2(F_i)$$

Suppose the words of the previous example were given in that order: /imaZ/ - /vwAtyR/ - /SosyR/ - /imaZ/ - /plaZ/ - /SosyR/ - /plaZ/ - /imaZ/ - /etaZ/ - /vwAtyR/ - /baRaZ/ - /azyR/ - /plaZ/ - /imaZ/ - /imaZ/ - /kuRaZ/

Candidate set #2 would then have an average description length of 1.875 bits:

```

azyR m          -1/16 x log2(1/16) = .25
*yR f SosyR,vwAtyR -4/16 x log2(4/16) = .5
*RaZ m baRaZ,kuRaZ -2/16 x log2(2/16) = .375
etaZ m          -1/16 x log2(1/16) = .25
*aZ f imaZ,plaZ  -8/16 x log2(8/16) = .5

Sum = 1.875 bits

```

In the same way, candidate set #1 would have a value of 2.18 bits. Candidate set #3 would have a value of 2 bits. The best model is therefore model #2 which is the most compact one, according to the word frequencies.

4 Implementation

For computational purposes, the knowledge internal representation is slightly different than the one we use here: rules and exceptions are represented on different lines such that exceptions are written before their corresponding rules and if a rule is more specific than another one, it is written before. For instance, candidate set #2 is written that way:

```

azyR m
*yR f SosyR,vwAtyR
*RaZ m baRaZ,kuRaZ
etaZ m
*aZ f imaZ,plaZ

```

This allows a linear inspection of the rule base in order to predict the gender of a new word: the first rule which matches the new word gives the gender. For instance, if the previous model were selected, it would predict that the word /caZ/ is feminine, the pseudo-word /tapyR/ is feminine and the pseudo-word /piRaZ/ is masculine.

We could have improved the efficiency of the algorithm by organizing words in a prefix tree where the keys would be in the reverse order of words. However, we are not concerned with the efficiency of the model for the moment, but rather its ability to account for human data.

The algorithm is the following ($R_1 < R_2$ indicates that R_1 is more specific than R_2 . For instance, */tyR/ is more specific than */yR/, which in turn is more specific than */R/).

```

updateModel(word W, rule base B):
if W matches a rule R ∈ B then
  if R did not contain W as an example
    add W to the examples of B
  return B
else
  for all exceptions E of B
    if E and W can be generalized

```

```

create the new rule N from them
include possible other exceptions

```

```

# More general rule of different gender
if ∃R ∈ B/ R < N and gender(R) ≠ gender(N)
  put examples of N matching R as exceptions
  memorize those exceptions
  if N now contains one example
    put that example as an exception
  if N contains no examples
    remove N

```

```

# More specific rule of different gender
if ∃R ∈ B/ R ≥ N and gender(R) ≠ gender(N)
  put examples of R matching N as exceptions
  memorize those exceptions
  if R now contains one example
    put that example as an exception
  if R contains no examples
    remove R

```

```

# Conflicting rule of same gender
if ∃R ∈ B/ N > R and gender(R) = gender(N)
  include R into N
if ∃R ∈ B/ N < R and gender(R) = gender(N)
  include N into R

```

Solutions = {B}

```

# Run the algorithm with new exceptions
for all memorized exceptions E
  Solutions = Solutions ∪ updateModel(E,B)

```

```

if no generalizations was possible
  Add W to B
  Solutions = {B}

```

return(Solutions)

5 Simulations

We ran this model on two corpora, representing words grade-1 and grade-2 children are exposed to (each 200,000-word long). 76 rules were obtained in running the grade-1 corpus, and 83 rules with the grade-2 corpus.

End-ings	Gen-der	Gender Predict-ability	Nb Exam-ples	Nb excep-tions
*/l/	f	56%	79	62
*/sol/	m	57%	4	3
*/i/	m	57%	74	55
*/R/	m	72%	188	71
*/am/	f	77%	7	2
*/sy/	m	83%	5	1
*/jER/	f	88%	31	4
*/S/	m	97%	91	2
*/fon/	m	100%	5	0
*/sj6/	f	100%	58	0

Table 1. Sample of rules (with endings and predicted gender) constructed from grade-1 corpus.

Some of the rules of the first set are listed in Table I (from grade-1 corpus). For each rule, represented by a word ending, is detailed its predicted gender, the number of words (as types) following the rule, the number of exceptions. Moreover, the “gender predictability” of each rule is computed (third column) as the percentage of words matching the rule over the total number of words with this ending.

The results of the simulations show that the lengths of word endings vary from only one phoneme (e.g., /**l*/, /**i*/) to three (/**jER*/, /**fon*/). These rules do not really correspond to the kind of rules linguists would have produced. They usually consider that the appropriate ending to associate to a given gender is the suffix (Riegel et al., 2005). Actually, the nature of the word ending that humans may rely on to predict gender is an open question in psycholinguistics. Do we rely on the suffix, the last morpheme, the last phoneme? The results of our model which did not use any morphological knowledge, suggests another answer: it may only depend on the statistical regularities of word endings in the language and can vary in French from one phoneme to three and these endings are sometimes matching morphological units.

However, it is worth noting that the model has yet some obvious limitations. The first one is that the gender predictability of rules is variable: while some rules are highly predictive (e.g., **/sjʃ/* 100% feminine, **/@/* 97% masculine), other are not (e.g., **/l/* 56% feminine, **/i/* 57% masculine). The second limitation is that the rules found by our model are accounting for a variable amount of examples. For instance, the rule **/R/* masculine accounts for 188 examples while **/sol/* masculine does only 4. One could wonder what it means from a developmental point of view to create rules that are extracted from very few examples. Do children build such rules? This is far from sure and we shall have to further address these clear limitations.

Another of our research goals was to test to what extent our model could predict human data. To that end, the model’s gender assignment performance was compared to children’s one.

6 Comparison to Experimental Data

6.1 Experiment

An experiment was conducted to study how and when French native children acquire regularities between words endings and their associated gender. Nine endings were selected, five which are

more likely associated to the feminine gender (/ad/, /asjʃ/, /El/, /ot/, /tyR/) and four to the masculine gender (/aZ/, /m@/, /waR/, /O/). Two lists of 30 pseudo-words were created containing each 15 pseudo-words whose expected gender is masculine (such as “*brido*” or “*rinloir*”) and 15 whose expected gender is feminine (such as “*surbelle*” or “*marniture*”). The presentation of each list was counterbalanced across participants.

Participants were 136 children from Grenoble (all French native speakers): 28 children at the end of preschool, 30 children at the beginning of grade 1, 36 children at the end of grade 1 and 42 children at the beginning of grade 2. Each participant was given a list and had to perform a computer-based gender decision task. Each pseudo-word was simultaneously spoken and displayed in the center of the screen when the determiners “*le*” (masculine) and “*la*” (feminine) were displayed at the bottom of the screen. Then children had to press the keyboard key corresponding to their intuition, which was recorded.

End-ings	Gd.	Pre-school	Beg. Grade1	End Grade1	Beg. Grade2
		% Exp. Gd.	% Exp. Gd.	% Exp. Gd.	% Exp. Gd.
/ad/	f	45.24	56.67	67.59**	57.14
/asjʃ/	f	58.33	58.89	70.37**	65.08**
/El/	f	60.71*	62.22*	76.85**	64.29**
/ot/	f	53.57	71.11**	82.41**	72.22**
/tyR/	f	50.00	68.89**	77.78**	68.25**
/aZ/	m	51.19	64.44**	64.81**	61.11**
/m@/	m	60.71*	55.56	57.41	50.00
/O/	m	61.90*	65.56**	80.56**	78.57**
/waR/	m	52.38	62.22*	64.81**	68.25**

Legend: Gd.:Gender; Beg.:Beginning;
% Exp. Gd.:% Expected Gender;
* p<.05,**p<.01

Table 2. Gender attribution rate as a function of endings and grade level.

In brief, results are twofold. First, children have acquired some implicit knowledge regarding gender information associated with word ending. As can be seen in Table 2, at the beginning of grade 1, children respond above chance and in the expected direction for the majority of endings (Chi2 test was used to assess statistical significance). At preschool children responded also above chance for three word endings. Second, there is a clear developmental trend since gender attribution increases in the expected direction with grade level and more endings are determined by the older children. The exposure

to written language during the first school year probably reinforces the implicit knowledge developed by children before primary school.

6.2 Human vs. Model Data Comparison

Two types of analyses were drawn in order to compare model and data. Firstly, the gender predictions obtained from the model were correlated to those given by children, regarding the gender of pseudo-words. Secondly, the endings created by the model were compared to those used in the experimental material. Correlations were computed between our model and human data (Table 3) by taking into account the rate of predicted masculine gender, for each pseudo-word.

	Model Grade 1	Model Grade 2
Preschool	0.31	0.33
Beg. Grade 1	0.6	0.64
End Grade 1	0.82	0.86
Beg. Grade 2	0.74	0.77

Table 3. Correlations between model and data.

The highest correlations are obtained for children at the end of grade 1 and at the beginning of grade 2. This result is interesting since the corpora are precisely intended to represent the lexical knowledge corresponding to the school level of these children. Moreover, the correlations obtained with the grade-2 model are higher (though not significantly) than those obtained with the grade-1 model. It thus seems that our model is fairly well suited to account for children's results, at least for the older ones. The low correlations observed with the younger children of our sample cannot be interpreted unambiguously; one could say that children before grade 1 have not built much knowledge regarding gender of word endings but this conclusion contradicts previous results (Meunier et al., 2008) and it remains to be explored by using a corpora appropriated to the lexicon of preschool children.

The endings used by the model to predict the gender of pseudo-words were also compared with the endings used in the experiment. Table 4 presents these endings as well as the rate of masculine gender predicted for the experimental endings by the two models trained with grade-1 and grade-2 lexicons. First, note that the endings used by the models are the same for both grade-1 and grade-2 lexicons. The growth of the lexicon between grade 1 and grade 2 does not modify these rules. Secondly, one can notice that grade-2 model results are more defined than grade-1 re-

sults. Third, a very salient result is that model endings are short. For example, the model did not create a rule such **/ad/* and rather used the more compact rule **/d/* to predict the gender of the pseudo-word */bOSad/*.

Endings	Model Grade 1		Model Grade 2	
	End-ings	% Gd. Masc	End-ings	% Gd. Masc
<i>/ad/</i>	<i>*/d/</i>	0.28	<i>*/d/</i>	0.17
<i>/asj\$/</i>	<i>*/sj\$/</i>	0	<i>*/sj\$/</i>	0
<i>/El/</i>	<i>*/l/</i>	0.44	<i>*/l/</i>	0.32
<i>/ot/</i>	<i>*/t/</i>	0.14	<i>*/t/</i>	0.09
<i>/tyR/</i>	<i>*/yR/</i>	0.09	<i>*/yR/</i>	0.05
<i>/aZ/</i>	<i>*/Z/</i>	0.8	<i>*/Z/</i>	0.91
<i>/m@/</i>	<i>*/@/</i>	0.95	<i>*/@/</i>	0.98
<i>/O/</i>	<i>*/O/</i>	0.93	<i>*/O/</i>	0.96
<i>/waR/</i>	<i>*/R/</i>	0.72	<i>*/R/</i>	0.82

Table 4. Rate for expected masculine gender predicted by our models.

In fact, the majority of the endings used by the model are short, i.e. composed with one phoneme. Very few endings created by the model are morphological units such as suffixes. In fact, the endings */d/* or */R/* are not derivational morphemes, but the endings */sj\$/* or */yR/* are suffixes. So the MDL-based model establishes rules that take into account different types of linguistic units from phonemes to morphemes depending of the statistical predictability of each ending type. This result is related to an important concern about the study of the acquisition of grammatical gender: to which unit do children rely on to predict gender? Do they rely on the last phoneme, biphone, morpheme?

7 Do children rely on morphemes?

In grammatical gender acquisition studies, the kind of endings used often mixes up phonological, derivational and even orthographic cues. Several studies used true suffixes (Marchal et al., 2007, Meunier et al., 2008) to ask children to assign gender to pseudo-words. As those studies consistently showed that children from 3 years old onwards assign a gender to those pseudo-words following the expected suffix gender, the tentative conclusion was to say that children rely on suffixes to assign the gender of new words. This is an appealing interpretation as the development of morphological structure of words is an important aspect of lexical development and some of this knowledge is acquired very early (Casalis et al., 2000; Karmiloff-Smith, 1979).

However, the observations from the MDL-based model strongly question this assumption: the units retained in the model's rules are often shorter than suffixes and the last phoneme seems often as predictive as the suffix itself as it leads to satisfying correlations with children's data.

So, one would conclude that gender knowledge is not attached to morphological units such as suffix but is rather a knowledge associated with the smaller ending segment that best predicts gender. Note however that despite the high correlations observed, the actual gender predictions issued from children's data and those issued from the model are not exactly of the same magnitude and this would suggest that the MDL-based model presented here must still be worked on in order to better describe gender acquisition. For example, the notion of gender predictability would benefit from being computed from token counts instead of type counts.

8 Conclusion

The purpose of this research was to know which kind of gender information may be constructed and used by children, and which cognitive mechanisms may lead to the construction of such rules. To investigate that issue, we constructed a model based on the MDL principle which reveals to be an interesting way to describe the grammatical gender acquisition in French, although we do not claim that children employ such an algorithm. Our model predicts the gender of a new word by sequentially scanning exceptions and rules. This process appears quite similar to the decision lists technique in machine learning (Rivest, 1987) which has already been combined with the MDL principle (Pfahring, 1997). However, we are not committed to this formalism: we are more interested in the content of the model rather than its knowledge representation. The comparison between model's results and human data opens a way of reflection on the kind of relevant units on which children would rely on. Perhaps it is not a kind of ending in particular that plays a role but different units varying following the principle of parsimony.

References

Casalis, S., Louis-Alexandre, M.-F. (2000). Morphological analysis, phonological analysis and learning to read French. *Reading and Writing*, 12, 303-335.

Chater, N. (1999). The search for simplicity: A fundamental cognitive principle? *Quarterly Journal of Experimental Psychology*, 52A, 273-302.

Chater, N., & Vitányi, P. (2003). Simplicity: a unifying principle in cognitive science? *Trends in Cognitive Sciences*, 7(1), 19-22.

Chater, N., & Vitanyi, P. (2007) 'Ideal learning' of natural language: Positive results about learning from positive evidence. *Journal of Mathematical Psychology*, 51(3), 135-163.

Feldman, J. (2003). Perceptual Grouping by Selection of a Logically Minimal Model. *International Journal of Computer Vision*, 55(1), 5-25.

Grünwald, P. (2005). Minimum description length tutorial. In P. D. Grünwald, I. J. Myung & M. Pitt (Eds.), *Advances in MDL: Theory and Applications* (pp. 23-80). Cambridge: MIT Press.

Holmes, V.M., & Segui, J. (2004). Sublexical and lexical influences on gender assignment in French. *Journal of Psycholinguistic Research*, 33(6), 425-457.

Karmiloff-Smith, A. (1979). *A functional approach to child language*. Cambridge University Press.

Lété, B., Sprenger-Charolles, L., & Colé, P. (2004). MANULEX: A grade-level lexical database from French elementary-school readers. *Behavior Research Methods, Instruments, & Computers*, 36, 156-166.

Marchal, H., Bianco, M., Dessus, P. & Lemaire, B. (2007). The Development of Lexical Knowledge: Toward a Model of the Acquisition of Lexical Gender in French. *Proceedings of the 2nd European Conference on Cognitive Science*, 268-273.

Matthews, C. A. (2005). French gender attribution on the basis of similarity: A comparison between AM and connectionist models. *Journal of Quantitative Linguistics*, 12(2-3), 262-296.

Meunier, F., Seigneuric, A., Spinelli, E. (2008). The morpheme gender effect. *Journal of Memory and Language*, 58, 88-99.

Pfahring, B. (1997). Compression-Based Pruning of Decision Lists, in *Proceedings of the 9th European Conference on Machine Learning*, 199-212.

Riegel, M., Pellat, J.C., & Rioul, R. (2005). *Grammaire méthodique du français*. Paris: PUF.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14, 465-471.

Rivest, R.L. (1987). Learning Decision Lists. *Machine Learning* 2,3 (1987), 229-246.

Skousen, R. (2003). *Analogical Modeling: Exemplars, Rules, and Quantum Computing*. Berkeley Linguistics Society.

Baby SRL: Modeling Early Language Acquisition.

Michael Connor

Department of Computer Science
University of Illinois
connor2@uiuc.edu

Yael Gertner

Beckman Institute
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

A fundamental task in sentence comprehension is to assign semantic roles to sentence constituents. The structure-mapping account proposes that children start with a shallow structural analysis of sentences: children treat the number of nouns in the sentence as a cue to its semantic predicate-argument structure, and represent language experience in an abstract format that permits rapid generalization to new verbs. In this paper, we tested the consequences of these representational assumptions via experiments with a system for automatic semantic role labeling (SRL), trained on a sample of child-directed speech. When the SRL was presented with representations of sentence structure consisting simply of an ordered set of nouns, it mimicked experimental findings with toddlers, including a striking error found in children. Adding features representing the position of the verb increased accuracy and eliminated the error. We show the SRL system can use incremental knowledge gain to switch from error-prone noun order features to a more accurate representation, demonstrating a possible mechanism for this process in child development.

1 Introduction

How does the child get started in learning to interpret sentences? The structure-mapping view of early verb and syntax acquisition proposes that

children start with a shallow structural analysis of sentences: 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 permits rapid generalization to new verbs (Gertner et al., 2006).

The structure-mapping account makes strong predictions. First, as soon as children can identify some nouns, they should interpret transitive and intransitive sentences differently, simply by assigning a distinct semantic role to each noun in the sentence. Second, language-specific syntactic learning should transfer rapidly to new verbs. Third, some striking errors of interpretation can occur. In “Fred and Ginger danced”, an intransitive verb is presented with two nouns. If children interpret any two-noun sentence as if it were transitive, they should be fooled into interpreting the order of two nouns in such conjoined-subject intransitive sentences as conveying agent-patient role information. Experiments with young children support these predictions. First, 21-month-olds use the number of nouns to understand sentences containing new verbs (Yuan et al., 2007). Second, 21-month-olds generalize what they have learned about English transitive word-order to sentences containing new verbs: Children who heard “The girl is gorging the boy” interpreted the girl as an agent and the boy as a patient (Gertner et al., 2006). Third, 21-month-olds make the predicted error, treating intransitive sentences containing two nouns as if they were transitive: they interpret the first noun in “The girl and the boy are gorging” as an agent and the second as a patient (Gertner and Fisher, 2006). This error is short-lived. By 25 months, children add new features to their representations of sentences, and interpret conjoined-subject intransitives differ-

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

ently from transitives (Naigles, 1990).

These experimental results shed light on what syntactic information children might have available for early sentence comprehension, but do not rule out the possibility that children’s early performance is based on a more complex underlying system. In this paper, we tested the consequences of our representational assumptions by performing experiments with a system for automatic semantic role labeling (SRL), whose knowledge of sentence structure is under our control. Computational models of semantic role labeling learn to identify, for each verb in a sentence, all constituents that fill a semantic role, and to determine their roles. We adopt the architecture proposed by Roth and colleagues (Punyakanok et al., 2005), limiting the classifier’s features to a set of lexical features and shallow structural features suggested by the structure-mapping account. Learning ability is measured by the level of SRL accuracy and, more importantly, the types of errors made by the system on sentences containing novel verbs. Testing these predictions on the automatic SRL provides us with a demonstration that it is possible to learn how to correctly assign semantic roles based only on these very simple cues.

From an NLP perspective this feature study provides evidence for the efficacy of alternative, simpler syntactic representations in gaining an initial foothold on sentence interpretation. It is clear that human learners do not begin interpreting sentences in possession of full part-of-speech tagging, or full parse trees. By building a model that uses shallow representations of sentences and mimics features of language development in children, we can explore the nature of initial representations of syntactic structure and build more complex features from there, further mimicking child development.

2 Learning Model

We trained a simplified SRL classifier (Baby SRL) with sets of features derived from the structure-mapping account. Our test used novel verbs to mimic sentences presented in experiments with children. Our learning task is similar to the full SRL task (Carreras and Márquez, 2004), except that we classify the roles of individual words rather than full phrases. A full automatic SRL system (e.g. (Punyakanok et al., 2005)) 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 SRL architecture (Baby SRL) essentially replaces the first two steps with heuristics. Rather than identifying arguments via a learned classifier with access to a full syntactic parse, the Baby SRL 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 Baby SRL 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.

Feedback is provided based on annotation in Propbank style: in training, each noun receives the role label of the phrase that noun is part of. Feedback is given at the level of the macro-role (agent, patient, etc., labeled A0-A4 for core arguments, and AM-* adjuncts). We also introduced a NO label for nouns that are not part of any argument.

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 has been demonstrated to work well in other text classification tasks, and allows us to inspect the weights of key features to determine their importance for classification. The Baby SRL does not use inference for the final classification. Instead it classifies every argument independently; thus multiple nouns can have the same role.

2.1 Training

The training data were samples of parental speech to one child (‘Eve’; (Brown, 1973), available via Childes (MacWhinney, 2000)). We trained on parental utterances in samples 9 through 20, recorded at child age 21-27 months. 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 system (Punyakanok et al., 2005). In this initial pass, sentences with parsing errors that misidentified argument boundaries were excluded. Final role labels were hand-corrected using the Propbank annotation scheme (Kingsbury and Palmer, 2002). The child-directed speech (CDS) training set consisted of about 2200 sentences, of which a majority had a single verb and two nouns to be labeled¹. We used the annotated CDS training data to train our Baby SRL, converting labeled phrases to labeled nouns in the manner described above.

3 Experimental Results

To evaluate the Baby SRL we tested it with sentences like those used for the experiments with children described above. All 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. We filled the A and B slots by sampling nouns that occurred roughly equally as the first and second of two nouns in the training data. This procedure was adopted to avoid 'building in' the predicted error by choosing A and B nouns biased toward an agent-patient interpretation. For each test sentence template we built a test set of 100 sentences by randomly sampling nouns in this fashion.

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 the patient (A1)? Does it overgeneralize this rule to two-noun intransitives ('A and B gorp'), mimicking children's behavior? We used two measures of success, one to assess classification accuracy, and the other to assess the predicted error. We used a per argument F1 for classification accuracy, with F1 based on correct identification of individual nouns rather than full phrases. Here precision is defined as the proportion of nouns that were given the correct label based on the argument they belong to, and recall is the proportion of complete arguments for which

some noun in that argument was correctly labeled. The desired labeling for 'A gorps B' is A0 for the first argument and A1 for the second; for 'A and B gorp' both arguments should be A0. To measure predicted errors we also report the proportion of test sentences classified with A0 first and A1 second (%A0A1). This labeling is a correct generalization for the novel 'A gorps B' sentences, but is an overgeneralization for 'A and B gorp.'

3.1 Noun Pattern

The basic feature we propose is the noun pattern feature. We hypothesize that children use the number and order of nouns to represent argument structure. To encode this we created a feature (NPattern) that indicates how many nouns there are in the sentence and which noun the target is. For example, in our two-noun test sentences noun A has the feature '_N' active indicating that it is the first noun of two. Likewise for B the feature 'N_' 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 distinctions between types of nouns or any other part of speech. Table 1 shows the initial feature progression that involves this feature. The baseline system (feature set 1) uses lexical features only: the target noun and the root form of the predicate.

We first tested the hypothesis that children use the NPattern features to distinguish different noun arguments, but only for specific verbs. The NPattern&V features are conjunctions of the target verb and the noun pattern, and these are added to the word features to form feature set 2. Now every example has three features active: target noun, target predicate, and a NPattern&V feature indicating 'the target is the first of two nouns and the verb is X.' This feature does not improve results on the novel 'A gorps B' test set, or generate the predicted error with the 'A and B gorp' test set, because the verb-specific NPattern&V features provide no way to generalize to unseen verbs.

We next tested the NPattern feature alone, without making it verb-specific (feature set 3). The noun pattern feature was added to the word features and again each example had three features active: target noun, target predicate, and the target's noun-pattern feature (first of two, second of three, etc.). The abstract NPattern feature allows the Baby SRL to generalize to new verbs: it increases the system's tendency to predict that the first of two

¹Corpus available at <http://L2R.cs.uiuc.edu/~cogcomp/data.php>

Features	CHILDES								WSJ			
	Unbiased Noun Choice				Biased Noun Choice				Biased Noun Choice			
	A gorps B		A and B gorp		A gorps B		A and B gorp		A gorps B		A and B gorp	
	F1	%A0A1	F1	%A0A1	F1	%A0A1	F1	%A0A1	F1	%A0A1	F1	%A0A1
1. Words	0.59	0.38	0.46	0.38	0.80	0.65	0.53	0.65	0.57	0.31	0.37	0.31
2. NPattern&V	0.53	0.28	0.54	0.28	0.81	0.67	0.53	0.67	0.56	0.31	0.39	0.31
3. NPattern	0.83	0.65	0.33	0.65	0.96	0.92	0.46	0.92	0.67	0.44	0.37	0.44
4. NPattern + NPattern&V	0.83	0.65	0.33	0.65	0.95	0.90	0.45	0.90	0.73	0.53	0.44	0.53
5. + VPosition	0.99	0.96	0.98	0.00	1.00	1.00	0.99	0.01	0.94	0.88	0.69	0.39

Table 1: Experiments showing the efficacy of Noun Pattern features for determining agent/patient roles in simple two-noun sentences. The novel verb test sets assess whether the Baby SRL generalizes transitive argument prediction to unseen verbs in the case of ‘A gorps B’ (increasing %A0A1 and thus F1), and overgeneralizes in the case of ‘A and B gorp’ (increasing %A0A1, which is an error). By varying the sampling method for creating the test sentences we can start with a biased or unbiased lexical baseline, demonstrating that the noun pattern features still improve over knowledge that can be contained in typical noun usage. The simple noun pattern features are still effective at learning this pattern when trained with more complex Wall Street Journal training data.

nouns is A0 and the second of two nouns is A1 for verbs not seen in training. Feature set 4 includes both the abstract, non-verb-specific NPattern feature and the verb-specific version. This feature set preserves the ability to generalize to unseen verbs; thus the availability of the verb-specific NPattern features during training did not prevent the abstract NPattern features from gathering useful information.

3.2 Lexical Cues for Role-Labeling

Thus far, the target nouns’ lexical features provided little help in role labeling, allowing us to clearly see the contribution of the proposed simple structural features. Would our structural features produce any improvement above a more realistic lexical baseline? We created a new set of test sentences, sampling the A nouns based on the distribution of nouns seen as the first of two nouns in training, and the B nouns based on the distribution of nouns seen as the second of two nouns. Given this revised sampling of nouns, the words-only baseline is strongly biased toward A0A1 (biased results for feature set 1 in table 1). This high baseline reflects a general property of conversation: Lexical choices provide considerable information about semantic roles. For example, the 6 most common nouns in the Eve corpus are pronouns that are strongly biased in their positions and in their semantic roles (e.g., ‘you’, ‘it’). Despite this high baseline, however, we see the same pattern in the unbiased and biased experiments in table 1. The addition of the NPattern features (feature set 3) substantially improves performance on ‘A gorps B’ test sentences, and promotes overgeneralization errors on ‘A and B gorp’ sentences.

3.3 More Complex Training Data

For comparison purposes we also trained the Baby SRL on a subset of the Propbank training data of Wall Street Journal (WSJ) text (Kingsbury and Palmer, 2002). To approximate the simpler sentences of child-directed speech we selected only those sentences with 8 or fewer words. This provided a training set of about 2500 sentences, most with a single verb and two nouns to be labeled. The CDS and WSJ data pose similar problems for learning abstract and verb-specific knowledge. However, newspaper text differs from casual speech to children in many ways, including vocabulary and sentence complexity. One could argue that the WSJ corpus presents a worst-case scenario for learning based on shallow representations of sentence structure: Full passive sentences are more common in written corpora such as the WSJ than in samples of conversational speech, for example (Roland et al., 2007). As a result of such differences, two-noun sequences are less likely to display an A0-A1 sequence in the WSJ (0.42 A0-A1 in 2-noun sentences) than in the CDS training data (0.67 A0-A1). The WSJ data provides a more demanding test of the Baby SRL.

We trained the Baby SRL on the WSJ data, and tested it using the biased lexical choices as described above, sampling A and B nouns for novel-verb test sentences based on the distribution of nouns seen as the first of two nouns in training, and as the second of two nouns, respectively. The WSJ training produced performance strikingly similar to the performance resulting from CDS training (last 4 columns of Table 1). Even in this more complex training set, the addition of the NPattern

features (feature set 3) improves performance on 'A gorps B' test sentences, and promotes over-generalization errors on 'A and B gorp' sentences.

3.4 Tests with Familiar Verbs

Features	Total	A0	A1	A2	A4
1. Words	0.64	0.83	0.74	0.33	0.00
2. NPattern&V	0.67	0.86	0.77	0.45	0.44
3. NPattern	0.66	0.87	0.76	0.37	0.22
4. NPattern + NPattern&V	0.68	0.87	0.80	0.47	0.44
5. + VPosition	0.70	0.88	0.83	0.50	0.50

Table 2: *Testing NPattern features on full SRL task of heldout section 8 of Eve when trained on sections 9 through 20. Each result column reflects a per argument F1.*

Learning to interpret sentences depends on balancing abstract and verb-specific structural knowledge. Natural linguistic corpora, including our CDS training data, have few verbs of very high frequency and a long tail of rare verbs. Frequent verbs occur with differing argument patterns. For example, 'have' and 'put' are frequent in the CDS data. 'Have' nearly always occurs in simple transitive sentences that display the canonical word order of English (e.g., 'I have cookies'). 'Put', in contrast, tends to appear in non-canonical sentences that do not display an agent-patient ordering, including imperatives ('Put it on the floor'). To probe the Baby SRL's ability to learn the argument-structure preferences of familiar verbs, we tested it on a held-out sample of CDS from the same source (Eve sample 8, approximately 234 labeled sentences). Table 2 shows the same feature progression shown previously, with the full SRL test set. The words-only baseline (feature set 1 in Table 2) yields fairly accurate performance, showing that considerable success in role assignment in these simple sentences can be achieved based on the argument-role biases of the target nouns and the familiar verbs. Despite this high baseline, however, we still see the benefit of simple structural features. Adding verb-specific (feature set 2) or abstract NPattern features (feature set 3) improves classification performance, and the combination of both verb-specific and abstract NPattern features (feature set 4) yields higher performance than either alone. The combination of abstract NPattern features with the verb-specific versions allows the Baby SRL both to generalize to unseen verbs, as seen in earlier sections, and to learn the idiosyncrasies of known verbs.

3.5 Verb Position

The noun pattern feature results show that the Baby SRL can learn helpful rules for argument-role assignment using only information about the number and order of nouns. It also makes the error predicted by the structure-mapping account, and documented in children, because it has no way to represent the difference between the 'A gorps B' and 'A and B gorp' test sentences. At some point the learner must develop more sophisticated syntactic representations that could differentiate these two. These could include many aspects of the sentence, including noun-phrase and verb-phrase morphological features, and word-order features. As a first step in examining recovery from the predicted error, we focused on word-order features. We did this by adding a verb position feature (VPosition) that specifies whether the target noun is before or after the verb. Now simple transitive sentences in training should support the generalization that pre-verbal nouns tend to be agents, and post-verbal nouns tend to be patients. In testing, the Baby SRL's classification of the 'A gorps B' and 'A and B gorp' sentences should diverge.

When we add verb position information (feature set 5 in table 1 and 2), performance improves still further for transitive sentences, both with biased and unbiased test sentences. Also, for the first time, the A0A1 pattern is predicted less often for 'A and B gorp' sentences. This error diminished because the classifier was able to use the verb position features to distinguish these from 'A gorps B' sentences.

Features	Unbiased Lexical			
	A gorps B		A and B gorp	
	F1	%A0A1	F1	%A0A1
1. Words	0.59	0.38	0.46	0.38
3. NPattern	0.83	0.65	0.33	0.65
6. VPosition	0.99	0.95	0.97	0.00

Table 3: *Verb Position vs. Noun Pattern features alone. Verb position features yield better overall performance, but do not replicate the error on 'A and B gorp' sentences seen with children.*

Verb position alone provides another simple abstract representation of sentence structure, so it might be proposed as an equally natural initial representation for human learners, rather than the noun pattern features we proposed. The VPosition features should also support learning and generalization of word-order rules for interpreting transitive sentences, thus reproducing some of

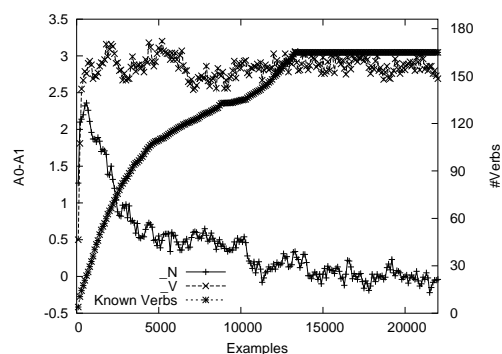
the data from children that we reviewed above. In table 3 we compared the words-only baseline (set 1), words and NPattern features (set 3), and a new feature set, words and VPosition (set 6). In terms of correct performance on novel transitive verbs ('A gorps B'), the VPosition features outperform the NPattern features. This may be partly because the same VPosition features are used in all sentences during training, while the NPattern features partition sentences by number of nouns, but is also due to the fact that the verb position features provide a more sophisticated representation of English sentence structure. Verb position features can distinguish transitive sentences from imperatives containing multiple post-verbal nouns, for example. Although verb position is ultimately a more powerful representation of word order for English sentences, it does not accurately reproduce a 21-month-old's performance on all aspects of this task. In particular, the VPosition feature does not support the overgeneralization of the A0A1 pattern to the 'A and B gorp' test sentences. This suggests that children's very early sentence comprehension is dominated by less sophisticated representations of word order, akin to the NPattern features we proposed.

3.6 Informativeness vs. Availability

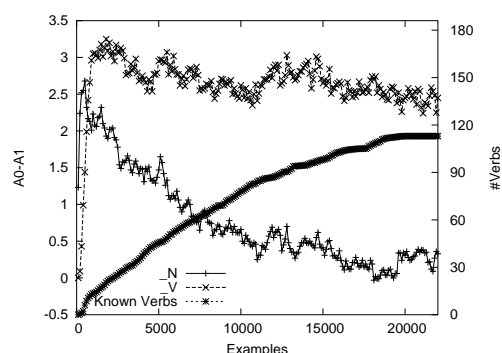
In the preceding sections, we modeled increases in syntactic knowledge by building in more sophisticated features. The Baby SRL escaped the predicted error on two-noun intransitive sentences when given access to features reflecting the position of the target noun relative to the verb. This imposed sequence of features is useful as a starting point, but a more satisfying approach would be to use the Baby SRL to explore possible reasons why NPattern features might dominate early in acquisition, even though VPosition features are ultimately more useful for English.

In theory, a feature might be unavailable early in acquisition because of its computational complexity. For example, lexical features are presumably less complex than relative position features such as NPattern and VPosition. In practice, features can also be unavailable at first because of an informational lack. Here we suggest that NPattern features might dominate VPosition features early in acquisition because the early lexicon is dominated by nouns, and it is easier to compute position relative to a known word than to an unknown word. Many

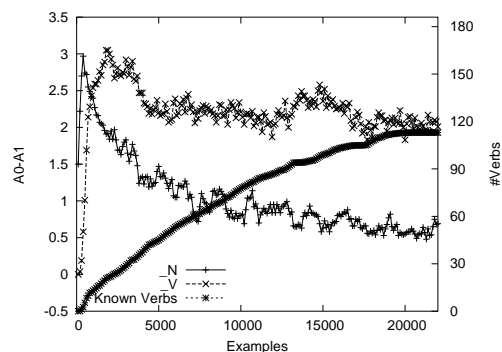
studies have shown that children's early vocabulary is dominated by names for objects and people (Gentner and Boroditsky, 2001).



(a) Verb threshold = 5



(b) Verb threshold = 20



(c) Verb threshold = 20, +verb-specific features

Figure 1: *Testing the consequences of the assumption that Verb Position features are only active for familiar verbs. The figure plots the bias of the features 'N' and 'V' to predict A0 over A1, as the difference between the weights of these connections in the learned network. Verb position features win out over noun pattern features as the verb vocabulary grows. Varying the verb familiarity threshold ((a) vs. (b)) and the presence versus absence of verb-specific versions of the structural features ((b) vs. (c)) affects how quickly the verb position features become dominant.*

To test the consequences of this proposed infor-

mational bottleneck on the relative weighting of NPattern and VPosition features during training, we modified the Baby SRL’s training procedure such that NPattern features were always active, but VPosition features were active during training only when the verb in the current example had been encountered a critical number of times. This represents the assumption that the child can recognize which words in the sentence are nouns, based on lexical familiarity or morphological context (Waxman and Booth, 2001), but is less likely to be able to represent position relative to the verb without knowing the verb well.

Figure 1 shows the tendency of the NPattern feature ‘_N’ (first of two nouns) and the VPosition feature ‘_V’ (pre-verbal noun) to predict the role A0 as opposed to A1 as the difference between the weights of these connections in the learned network. Figure 1(a) shows the results when VPosition features were active whenever the target verb had occurred at least 5 times; in Figure 1(b) the threshold for verb familiarity was 20. In both figures we see that the VPosition features win out over the NPattern features as the verb vocabulary grows. Varying the degree of verb familiarity required to accurately represent VPosition features affects how quickly the VPosition features win out (compare Figures 1(a) and 1(b)). Figure 1(c) shows the same analysis with a threshold of 20, but with verb-specific as well as abstract versions of the NPattern and the VPosition features. In this procedure, every example started with three features: target noun, target predicate, NPattern, and if the verb was known, added NPattern&V, VPosition, and VPosition&V. Comparing Figures 1(b) and 1(c), we see that the addition of verb-specific versions of the structural features also affects the rate at which the VPosition features come to dominate the NPattern features.

Thus, in training the VPosition features become dominant as the SRL learns to recognize more verbs. However, the VPosition features are inactive when the Baby SRL encounters the novel-verb test sentences. Since the NPattern features are active in test, the system generates the predicted error until the bias of the NPattern features reaches 0. Note in figure 1(c) that when verb-specific structural features were added, the Baby SRL never learned to entirely discount the NPattern features within the range of training provided. This result is reminiscent of suggestions in the psycholinguis-

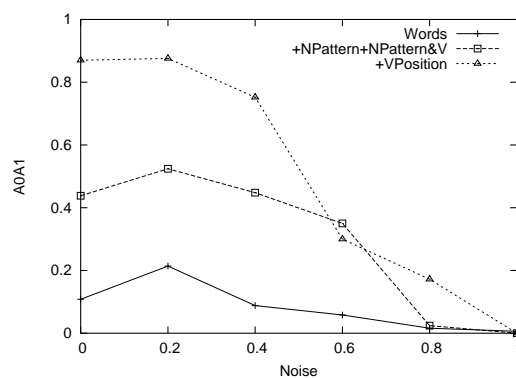


Figure 2: *Testing the ability of simple features to cope with varying amounts of noisy feedback. Even with noisy feedback, the noun pattern features support learning and generalization to new verbs of a simple agent-patient template for understanding transitive sentences. These results are lower than those found in table 1 due to slightly different training assumptions.*

tics literature that shallow representations of syntax persist in the adult parser, alongside more sophisticated representations (e.g., (Ferreira, 2003)).

3.7 Noisy Training

So far, the Baby SRL has only been trained with perfect feedback. Theories of human language acquisition assume that learning to understand sentences is naturally a partially-supervised task: the child uses existing knowledge of words and syntax to assign a meaning to a sentence; the appropriateness of this meaning for the referential context provides the feedback (e.g., (Pinker, 1989)). But this feedback must be noisy. Referential scenes provide useful but often ambiguous information about the semantic roles of sentence participants. For example, a participant could be construed as an agent of fleeing or as a patient being chased. In a final set of experiments, we examined the generalization abilities of the Baby SRL as a function of the integrity of semantic feedback.

We provided noisy semantic-role feedback during training by giving a randomly-selected argument label on 0 to 100% of examples. Following this training, we tested with the ‘A gorps B’ test sentences, using the unbiased noun choices.

As shown in Figure 2, feature sets including NPattern or VPosition features yield reasonable performance on the novel verb test sentences up to 50% noise, and promote an A0-A1 sequence over

the words-only baseline even at higher noise levels. Thus the proposed simple structural features are robust to noisy feedback.

4 Conclusion

The simplified SRL classifier mimicked experimental results with toddlers. We structured the learning task to ask whether shallow representations of sentence structure provided a useful initial representation for learning to interpret sentences. Given representations of the number and order of nouns in the sentence (noun pattern features), the Baby SRL learned to classify the first of two nouns as an agent and the second as a patient. When provided with both verb-general and verb-specific noun pattern features, the Baby SRL learned to balance verb-specific and abstract syntactic knowledge. By treating each noun as an argument, it also reproduced the errors children make. Crucially, verb-position features improved performance when added to the noun-pattern feature, but when presented alone failed to produce the error found with toddlers. We believe that our model can be naturally extended to support the case in which the arguments are noun phrases rather than single noun words and this extension is one of the first steps we will explore next.

Acknowledgments

We would like to thank our annotators, especially Yuancheng Tu. This research is supported by NSF grant BCS-0620257 and NIH grant R01-HD054448.

References

- Brown, R. 1973. *A First Language*. Harvard University Press, Cambridge, MA.
- Carreras, X. 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.
- Charniak, E. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proc. National Conference on Artificial Intelligence*.
- Ferreira, F. 2003. The misinterpretation of noncanonical sentences. *Cognitive Psychology*, 47:164–203.
- Fisher, C. 1996. Structural limits on verb mapping: The role of analogy in children’s interpretation of sentences. *Cognitive Psychology*, 31:41–81.
- Gentner, D. and L. Boroditsky. 2001. Individuation, relativity and early word learning. In Bowerman, M. and S. C. Levinson, editors, *Language acquisition and conceptual development*, pages 215–256. Cambridge University Press, New York.
- Gertner, Y. and C. Fisher. 2006. Predicted errors in early verb learning. In *31st Annual Boston University Conference on Language Development*.
- Gertner, Y., C. Fisher, and J. Eisengart. 2006. Learning words and rules: Abstract knowledge of word order in early sentence comprehension. *Psychological Science*, 17:684–691.
- Grove, A. and D. Roth. 2001. Linear concepts and hidden variables. *Machine Learning*, 42(1/2):123–141.
- Kingsbury, P. and M. Palmer. 2002. From Treebank to PropBank. In *Proceedings of LREC-2002*, Spain.
- MacWhinney, B. 2000. *The CHILDES project: Tools for analyzing talk. Third Edition*. Lawrence Erlbaum Associates, Mahwah, NJ.
- Naigles, L. R. 1990. Children use syntax to learn verb meanings. *Journal of Child Language*, 17:357–374.
- Pinker, S. 1989. *Learnability and Cognition*. Cambridge: MIT Press.
- Punyakanok, V., D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1117–1123.
- Roland, D., F. Dick, and J. L. Elman. 2007. Frequency of basic english grammatical structures: A corpus analysis. *Journal of Memory and Language*, 57:348–379.
- Waxman, S. R. and A. Booth. 2001. Seeing pink elephants: Fourteen-month-olds’s interpretations of novel nouns and adjectives. *Cognitive Psychology*, 43:217–242.
- Yuan, S., 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.

An Incremental Bayesian Model for Learning Syntactic Categories

Christopher Parisien, Afsaneh Fazly and Suzanne Stevenson

Department of Computer Science

University of Toronto

Toronto, ON, Canada

[chris, afsaneh, suzanne]@cs.toronto.edu

Abstract

We present an incremental Bayesian model for the unsupervised learning of syntactic categories from raw text. The model draws information from the distributional cues of words within an utterance, while explicitly bootstrapping its development on its own partially-learned knowledge of syntactic categories. Testing our model on actual child-directed data, we demonstrate that it is robust to noise, learns reasonable categories, manages lexical ambiguity, and in general shows learning behaviours similar to those observed in children.

1 Introduction

An important open problem in cognitive science and artificial intelligence is how children successfully learn their native language despite the lack of explicit training. A key challenge in the early stages of language acquisition is to learn the notion of abstract syntactic categories (e.g., nouns, verbs, or determiners), which is necessary for acquiring the syntactic structure of language. Indeed, children as young as two years old show evidence of having acquired a good knowledge of some of these abstract categories (Olguin and Tomasello, 1993); by around six years of age, they have learned almost all syntactic categories (Kemp et al., 2005). Computational models help to elucidate the kinds of learning mechanisms that may be capable of achieving this feat. Such studies shed light on the possible cognitive mechanisms at work in human language acquisition, and also on potential means for unsupervised learning of complex linguistic knowledge in a computational system.

Learning the syntactic categories of words has been suggested to be based on the morphological and phonological properties of individual words, as well

as on the distributional information about the contexts in which they appear. Several computational models have been proposed that draw on one or more of the above-mentioned properties in order to group words into discrete unlabeled categories. Most existing models only intend to show the relevance of such properties to the acquisition of adult-like syntactic categories such as nouns and verbs; hence, they do not necessarily incorporate the types of learning mechanisms used by children (Schütze, 1993; Redington et al., 1998; Clark, 2000; Mintz, 2003; Onnis and Christiansen, 2005). For example, in contrast to the above models, children acquire their knowledge of syntactic categories incrementally, processing the utterances they hear one at a time. Moreover, children appear to be sensitive to the fact that syntactic categories are partially defined in terms of other categories, e.g., nouns tend to follow determiners, and can be modified by adjectives.

We thus argue that a computational model should be incremental, and should use more abstract category knowledge to help better identify syntactic categories. Incremental processing also allows a model to incorporate its partially-learned knowledge of categories, letting the model *bootstrap* its development. To our knowledge, the only incremental model of category acquisition that also incorporates bootstrapping is that of Cartwright and Brent (1997). Their template-based model, however, draws on very specific linguistic constraints and rules to learn categories. Moreover, their model has difficulty with the variability of natural language data.

We address these shortcomings by developing an incremental probabilistic model of syntactic category acquisition that uses a domain-general learning algorithm. The model also incorporates a bootstrapping mechanism, and learns syntactic categories by looking only at the general patterns of distributional similarity in the input. Experiments performed on actual (noisy) child-directed data show that an explicit bootstrapping component improves the model's ability to

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

learn adult-like categories. The model’s learning trajectory resembles some relevant behaviours seen in children, and we also show that the categories that our model learns can be successfully used in a lexical disambiguation task.

2 Overview of the Computational Model

We adapt a probabilistic incremental model of unsupervised categorization (i.e., clustering) proposed by Anderson (1991). The original model has been used to simulate human categorization in a variety of domains, including the acquisition of verb argument structure (Alishahi and Stevenson, 2008). Our adaptation of the model incorporates an explicit bootstrapping mechanism and a periodic merge of clusters, both facilitating generalization over input data. Here, we explain the input to our model (Section 2.1), the categorization model itself (Section 2.2), how we estimate probabilities to facilitate bootstrapping (Section 2.3), and our approach for merging similar clusters (Section 2.4).

2.1 Input Frames

We aim to learn categories of words, and we do this by looking for groups of similar word usages. Thus, rather than categorizing a word alone, we categorize a word token *with* its context from that usage. The initial input to our model is a sequence of unannotated utterances, that is, words separated by spaces. Before being categorized by the model, each word usage in the input is processed to produce a *frame* that contains the word itself (the head word of the frame) and its distributional context (the two words before and after it). For example, in the utterance ‘I gave Josie a present,’ when processing the head word *Josie*, we create the following frame for input to the categorization system:

feature	w_{-2}	w_{-1}	w_0	w_{+1}	w_{+2}
	I	gave	Josie	a	present

where w_0 denotes the head word feature, and w_{-2} , w_{-1} , w_{+1} , w_{+2} are the context word features. A context word may be ‘null’ if there are fewer than two preceding or following words in the utterance.

2.2 Categorization

Using Anderson’s (1991) incremental Bayesian categorization algorithm, we learn clusters of word usages (i.e., the input frames) by drawing on the overall similarity of their features (here, the head word and the context words). The clusters themselves are not predefined, but emerge from similarities in the input. More formally, for each successive frame F in the input, processed in the order of the input words, we place F into the most likely cluster, either from the

K existing clusters, or a new one:

$$\text{BestCluster}(F) = \underset{k}{\operatorname{argmax}} P(k|F) \quad (1)$$

where $k = 0, 1, \dots, K$, including the new cluster $k = 0$. Using Bayes’ rule, and dropping $P(F)$ from the denominator, which is constant for all k , we find:

$$P(k|F) = \frac{P(k)P(F|k)}{P(F)} \propto P(k)P(F|k) \quad (2)$$

The prior probability of k , $P(k)$, is given by:

$$P(k) = \frac{cn_k}{(1-c) + cn}, \quad 1 \leq k \leq K \quad (3)$$

$$P(0) = \frac{1-c}{(1-c) + cn} \quad (4)$$

where n_k is the number of frames in k , and n is the total number of frames observed at the time of processing frame F . Intuitively, a well-entrenched (large) cluster should be a more likely candidate for categorization than a small one. We reserve a small probability for creating a new cluster (Eq. 4). As the model processes more input overall, it should become less necessary to create new clusters to fit the data, so $P(0)$ decreases with large n . In our experiments, we set c to a large value, 0.95, to further increase the likelihood of using existing clusters.¹

The probability of a frame F given a cluster k , $P(F|k)$, depends on the probabilities of the features in F given k . We assume that the individual features in a frame are conditionally independent given k , hence:

$$P(F|k) = P_H(w_0|k) \prod_{i \in \{-2, -1, +1, +2\}} P(w_i|k) \quad (5)$$

where P_H is the head word probability, i.e., the likelihood of seeing w_0 as a head word among the frames in cluster k . The context word probability $P(w_i|k)$ is the likelihood of seeing w_i in the i^{th} context position of the frames in cluster k . Next, we explain how we estimate each of these probabilities from the input.

2.3 Probabilities and Bootstrapping

For the head word probability $P_H(w_0|k)$, we use a smoothed maximum likelihood estimate (i.e., the proportion of frames in cluster k with head word w_0). For the context word probability $P(w_i|k)$, we can form two estimates. The first is a simple maximum likelihood estimate, which enforces a preference for creating clusters of frames with the same context words. That is, head words in the same cluster will

¹The prior $P(k)$ is equivalent to the prior in a Dirichlet process mixture model (Sanborn et al., 2006), commonly used for sampling clusters of objects.

tend to share the same adjacent words. We call this word-based estimate P_{word} .

Alternatively, we may consider the likelihood of seeing not just the context word w_i , but *similar* words in that position. For example, if w_i can be used as a noun or a verb, then we want the likelihood of seeing *other* nouns or verbs in position i of frames in cluster k . Here, we use the partial knowledge of the learned clusters. That is, we look over all existing clusters k' , estimate the probability that w_i is the head word of frames in k' , then estimate the probability of using the head words from those other clusters in position i in cluster k . We refer to this category-based estimate as P_{cat} :

$$P_{cat}(w_i|k) = \sum_{k'} P_H(w_i|k')P_i(k'|k) \quad (6)$$

where $P_i(k'|k)$ is the probability of finding usages from cluster k' in position i given cluster k . To support this we record the categorization decisions the model has made. When we categorize the frames of an utterance, we get a sequence of clusters for that utterance, which gives additional information to supplement the frame. We use this information to estimate $P_i(k'|k)$ for future categorizations, again using a smoothed maximum likelihood formula.

In contrast to the P_{word} estimate, the estimate in Eq. (6) prefers clusters of frames that use the same *categories* as context. While some of the results of these preferences will be the same, the latter approach lets the model make second-order inferences about categories. There may be no context words in common between the current frame and a potential cluster, but if the context words in the cluster have been found to be distributionally similar to those in the frame, it may be a good cluster for that frame.

We equally weight the word-based and the category-based estimates for $P(w_i|k)$ to get the likelihood of a context word; that is:

$$P(w_i|k) \approx \frac{1}{2}P_{word}(w_i|k) + \frac{1}{2}P_{cat}(w_i|k) \quad (7)$$

This way, the model sees an input utterance simultaneously as a sequence of words and as a sequence of categories. It is the P_{cat} component, by using developing category knowledge, that yields the bootstrapping abilities of our model.

2.4 Generalization

Our model relies heavily on the similarity of word contexts in order to find category structure. In natural language, these context features are highly variable, so it is difficult to draw consistent structure from the input in the early stages of an incremental model. When little information is available, there is a risk of

incorrectly generalizing, leading to clustering errors which may be difficult to overcome. Children face a similar problem in early learning, but there is evidence that they may manage the problem by using conservative strategies (see, e.g., Tomasello, 2000). Children may form specific hypotheses about each word type, only later generalizing their knowledge to similar words. Drawing on this observation, we form early small clusters specific to the head word type, then later aid generalization by merging these smaller clusters. By doing this, we ensure that the model only groups words of different types when there is sufficient evidence for their contextual similarity.

Thus, when a cluster has been newly created, we require that all frames put into the cluster share the same head word type.² When clusters are small, this prevents the model from making potentially incorrect generalizations to different words. Periodically, we evaluate a set of reasonably-sized clusters, and merge pairs of clusters that have highly similar contexts (see below for details). If the model decides to merge two clusters with different head word types—e.g., one cluster with all instances of *dog*, and another with *cat*—it has in effect made a decision to generalize. Intuitively, the model has learned that the contexts in the newly merged cluster apply to more than one word type. We now say that *any* word type could be a member of this cluster, if its context is sufficiently similar to that of the cluster. Thus, when categorizing a new word token (represented as a frame F), our model can choose from among the clusters with a matching head word, and any of these ‘generalized’ clusters that contain mixed head words.

Periodically, we look through a subset of the clusters to find similar pairs to merge. In order to limit the number of potential merges to consider, we only examine pairs of clusters in which at least one cluster has changed since the last check. Thus, after processing every 100 frames of input, we consider the clusters used to hold those recent 100 frames as candidates to be merged with another cluster. We only consider clusters of reasonable size (here, at least 10 frames) as candidates for merging. For each candidate pair of clusters, k_1 and k_2 , we first evaluate a heuristic merge score that determines if the pair is appropriate to be merged, according to some local criteria, i.e., the size and the contents of the candidate clusters. For each suggested merge (a pair whose merge score exceeds a pre-determined threshold), we then look at the set of all clusters, the *global* evidence, to decide whether to accept the merge.

The merge score combines two factors: the entrenchment of the two clusters, and the similarity of

²However, a word type may exist in several clusters (e.g., for distinct noun and verb usages), thus handling lexical ambiguity.

their context features. The entrenchment measure identifies clusters that contain enough frames to show a significant trend. We take a sigmoid function over the number of frames in the clusters, giving a soft threshold approaching 0 for small clusters and 1 for large clusters. The similarity measure identifies pairs of clusters with similar distributions of word and category contexts. Given two clusters, we measure the symmetric Kullback-Leibler divergence for each corresponding pair of context feature probabilities (including the category contexts $P_i(k'|k)$, 8 pairs in total), then place the sum of those measures on another sigmoid function. The merge score is the sum of the entrenchment and similarity measures.

Since it is only a local measure, the merge score is not sufficient on its own for determining if a merge is appropriate. For each suggested merge, we thus examine the likelihood of a sample of input frames (here, the last 100 frames) under two states: the set of clusters before the merge, and the set of clusters if the merge is accepted. We only accept a merge if it results in an increase in the likelihood of the sample data. The likelihood of a sample set of frames, \mathcal{S} , over a set of clusters, \mathcal{K} , is calculated as in:

$$P(\mathcal{S}) = \prod_{F \in \mathcal{S}} \sum_{k \in \mathcal{K}} P(F|k)P(k) \quad (8)$$

3 Evaluation Methodology

To test our proposed model, we train it on a sample of language representative of what children would hear, and evaluate its categorization abilities. We have multiple goals in this evaluation. First, we determine the model’s ability to discover adult-level syntactic categories from the input. Since this is intended to be a cognitively plausible learning model, we also compare the model’s qualitative learning behaviours with those of children. In the first experiment (Section 4), we compare the model’s categorization with a gold standard of adult-level syntactic categories and examine the effect of the bootstrapping component. The second experiment (Section 5) examines the model’s development of three specific parts of speech. Developmental evidence suggests that children acquire different syntactic categories at different ages, so we compare the model’s learning rates of nouns, verbs, and adjectives. Lastly, we examine our model’s ability to handle lexically ambiguous words (Section 6). English word forms commonly belong to more than one syntactic category, so we show how our model uses context to disambiguate a word’s category.

In all experiments, we train and test the model using the Manchester corpus (Theakston et al., 2001) from the CHILDES database (MacWhinney, 2000). The corpus contains transcripts of mothers’ conversations with 12 British children between the ages of

1;8 (years;months) and 3;0. There are 34 one-hour sessions per child over the course of a year. The age range of the children roughly corresponds with the ages at which children show the first evidence of syntactic categories.

We extract the mothers’ speech from each of the transcripts, then concatenate the input of all 12 children (all of Anne’s sessions, followed by all of Aran’s sessions, and so on). We remove all punctuation. We spell out contractions, so that each token in the input corresponds to only one part-of-speech (PoS) label (noun, verb, etc.). We also remove single-word utterances and utterances with a single repeated word type, since they contain no distributional information. We randomly split the data into development and evaluation sets, each containing approximately 683,000 tokens. We use the development set to fine-tune the model parameters and develop the experiments, then use the evaluation set as a final test of the model. We further split the development set into about 672,000 tokens (about 8,000 types) for training and 11,000 tokens (1,300 types) for validation. We split the evaluation set comparably, into training and test subsets. All reported results are for the evaluation set. A conservative estimate suggests that children are exposed to at least 1.5 million words of child-directed speech annually (Redington et al., 1998), so this corpus represents only a small portion of a child’s available input.

4 Experiment 1: Adult Categories

4.1 Methods

We use three separate versions of the categorization model, in which we change the components used to estimate the context word probability, $P(w_i|k)$ (as used in Eq. (5), Section 2.2). In the *word-based* model, we estimate the context probabilities using only the words in the context window, by directly using the maximum-likelihood P_{word} estimate. The *bootstrap* model uses only the existing clusters to estimate the probability, directly using the P_{cat} estimate from Eq. (6). The *combination* model uses an equally-weighted combination of the two probabilities, as presented in Eq. (7).

We run the model on the training set, categorizing each of the resulting frames in order. After every 10,000 words of input, we evaluate the model’s categorization performance on the test set. We categorize each of the frames of the test set as usual, treating the text as regular input. So that the test set remains unseen, the model does not record these categorizations.

4.2 Evaluation

The PoS tags in the Manchester corpus are too fine-grained for our evaluation, so for our gold standard

we map them to the following 11 tags: noun, verb, auxiliary, adjective, adverb, determiner, conjunction, negation, preposition, infinitive *to*, and ‘other.’ When we evaluate the model’s categorization performance, we have two different sets of clusters of the words in the test set: one set resulting from the gold standard, and another as a result of the model’s categorization. We compare these two clusterings, using the adjusted Rand index (Hubert and Arabie, 1985), which measures the overall agreement between two clusterings of a set of data points. The measure is ‘corrected for chance,’ so that a random grouping has an expected score of zero. This measure tends to be very conservative, giving values much lower than an intuitive percentage score. However, it offers a useful relative comparison of overall cluster similarity.

4.3 Results

Figure 1 gives the adjusted Rand scores of the three model variants, *word-based*, *bootstrap*, and *combination*. Higher values indicate a better fit with the gold-standard categorization scheme. The adjusted Rand score is corrected for chance, thus providing a built-in baseline measure. Since the expected score for a random clustering is zero, all three model variants operate at above-baseline performance.

As seen in Figure 1, the word-based model gains an early advantage in the comparison, but its performance approaches a plateau at around 200,000 words of input. This suggests that while simple word distributions provide a reliable source of information early in the model’s development, the information is not sufficient to sustain long-term learning. The bootstrap model learns much more slowly, which is unsurprising, given that it depends on having some reasonable category knowledge in order to develop its clusters—leading to a chicken-and-egg problem. However, once started, its performance improves well beyond the word-based model’s plateau. These results suggest that on its own, each component of the model may be effectively throwing away useful information. By combining the two models, the combination model appears to gain complementary benefits from each component, outperforming both. The word-based component helps to create a base of reliable clusters, which the bootstrap component uses to continue development.

After all of the training text, the combination model uses 411 clusters to categorize the test tokens (compared to over 2,000 at the first test point). While this seems excessive, we note that 92.5% of the test tokens are placed in the 25 most populated clusters.³

³See www.cs.toronto.edu/~chris/syncat for examples.

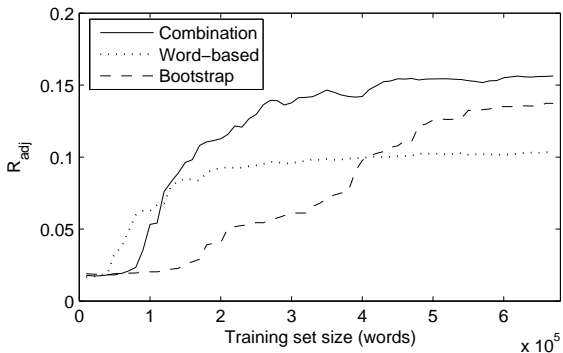


Figure 1: Adjusted Rand Index of each of three models’ clusterings of the test set, as compared with the PoS tags of the test data.

5 Experiment 2: Learning Trends

A common trend observed in children is that different syntactic categories are learned at different rates. Children appear to have learned the category of nouns by 23 months of age, verbs shortly thereafter, and adjectives relatively late (Kemp et al., 2005). Our goal in this experiment is to look for these specific trends in the behaviour of our model. We thus simulate an experiment where a child uses a novel word’s linguistic context to infer its syntactic category (e.g., Tomasello et al., 1997). For our experiment, we randomly generate input frames with novel head words using contexts associated with nouns, verbs, and adjectives, then examine the model’s categorization in each case. We expect that our model should approximate the developmental trends of children, who tend to learn the category of ‘noun’ before ‘verb,’ and both of these before ‘adjective.’

5.1 Methods

We generate new input frames using the most common syntactic patterns in the training data. For each of the noun, verb, and adjective categories (from the gold standard), we collect the five most frequent PoS sequences in which these are used, bounded by the usual four-word context window. For example, the Adjective set includes the sequence ‘V Det **Adj** N *null*’, where the sentence ends after the N. For each of the three categories, we generate each of 500 input frames by sampling one of the five PoS sequences, weighted by frequency, then sampling words of the right PoS from the lexicon, also weighted by frequency. We replace the head word with a novel word, forcing the model to use only the context for clustering. Since the context words are chosen at random, most of the word sequences generated will be novel. This makes the task more difficult, rather than simply sampling utterances from the corpus, where rep-

itions are common. While a few of the sequences may exist in the training data, we expect the model to mostly use the underlying category information to cluster the frames.

We intend to show that the model uses context to find the right category for a novel word. To evaluate the model’s behaviour, we let it categorize each of the randomly generated frames. We score each frame as follows: if the frame gets put into a new cluster, it earns score zero. Otherwise, its score is the proportion of frames in the chosen cluster matching the correct part of speech (we use a PoS-tagged version of the training corpus; for example, a noun frame put into a cluster with 60% nouns would get 0.6). We report the mean score for each of the noun, verb, and adjective sets. Intuitively, the matching score indicates how well the model recognizes that the given contexts are similar to input it has seen before. If the model clusters the novel word frame with others of the right type, then it has formed a category for the contextual information in that frame.

We use the full combination model (Eq. (7)) to evaluate the learning rates of individual parts of speech. We run the model on the training subset of the evaluation corpus. After every 10,000 words of input, we use the model to categorize the 1,500 context frames with novel words (500 frames each for noun, verb, and adjective). As in experiment 1, the model does not record these categorizations.

5.2 Results

Figure 2 shows the mean matching scores for each of the tested parts of speech. Recall that since the frames each use a novel head word, a higher matching score indicates that the model has learned to correctly recognize the contexts in the frames. This does not necessarily mean that the model has learned single, complete categories of ‘noun,’ ‘verb,’ and ‘adjective,’ but it does show that when the head word gives no information, the model can generalize based on the contextual patterns alone. The model learns to categorize novel nouns better than verbs until late in training, which matches the trends seen in children. Adjectives progress slowly, and show nearly no learning ability by the end of the trial. Again, this appears to reflect natural behaviour in children, although the effect we see here may simply be a result of the overall frequency of the PoS types. Over the entire corpus (development and evaluation), 35.4% of the word tokens are nouns and 24.3% are verbs, but only 2.9% are tagged as adjectives. The model, and similarly a child, may need much more data to learn adjectives than is available at this stage.

The scores in Figure 2 tend to fluctuate, particularly for the noun contexts. This fluctuation corresponds to periods of overgeneralization, followed

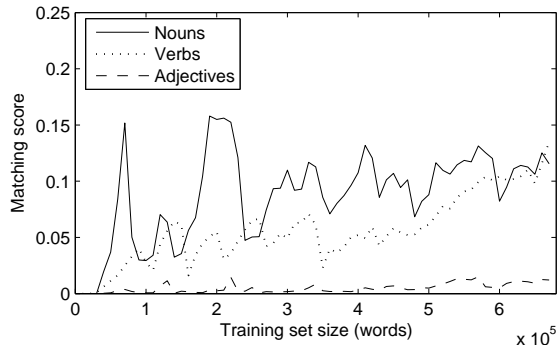


Figure 2: Comparative learning trends of noun, verb, and adjective patterns.

by recovery (also observed in children; see, e.g., Tomasello, 2000). When the model merges two clusters, the contents of the resulting cluster can initially be quite heterogeneous. Furthermore, the new cluster is much larger, so it becomes a magnet for new categorizations. This results in overgeneralization errors, giving the periodic drops seen in Figure 2. While our formulation in Section 2.4 aims to prevent such errors, they are likely to occur on occasion. Eventually, the model recovers from these errors, and it is worth noting that the fluctuations diminish over time. As the model gradually improves with more input, the dominant clusters become heavily entrenched, and inconsistent merges are less likely to occur.

6 Experiment 3: Disambiguation

The category structure of our model allows a single word type to be a member of multiple categories. For example, *kiss* could belong to a category of predominantly noun usages (*Can I have a kiss?*) and also to a category of verb usages (*Kiss me!*). As a result, the model easily represents lexical ambiguity. In this experiment, inspired by disambiguation work in psycholinguistics (see, e.g., MacDonald, 1993), we examine the model’s ability to correctly disambiguate category memberships.

6.1 Methods

Given a word that the model has previously seen as various different parts of speech, we examine how well the model can use that ambiguous word’s context to determine its category in the current usage. For example, by presenting the word *kiss* in separate noun and verb contexts, we expect that the model should categorize *kiss* as a noun, then as a verb, respectively. We also wish to examine the effect of the target word’s lexical bias, that is, the predominance of a word type to be used as one category over another. As with adults, if *kiss* is mainly used as a noun, we expect the model to more accurately categorize the

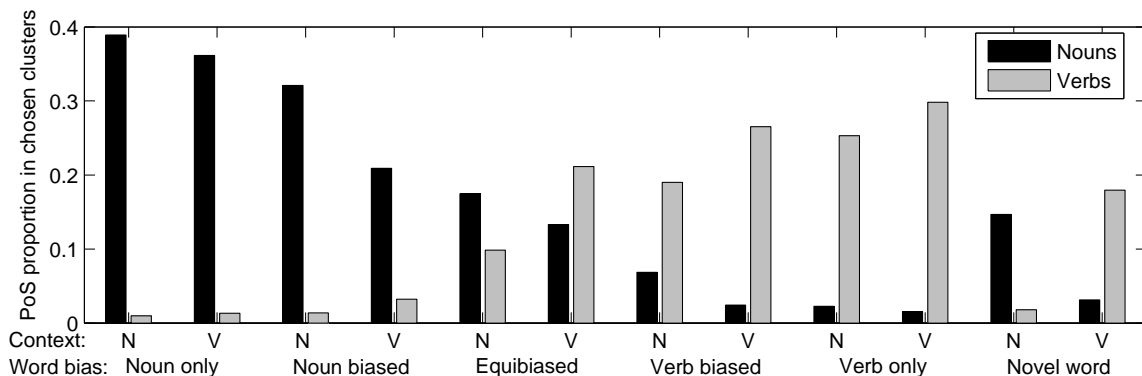


Figure 3: Syntactic category disambiguation. Shown are the proportions of nouns and verbs in the chosen clusters for ambiguous words used in either noun (N) or verb (V) contexts.

word in a noun context than in a verb context.

We focus on noun/verb ambiguities. We artificially generate input frames for noun and verb contexts as in experiment 2, with the following exceptions. To make the most use of the context information, we allow no *null* words in the input frames. We also ensure that the contexts are distinctive enough to guide disambiguation. For each PoS sequence surrounding a noun (e.g., ‘V Det **head** Prep Det’), we ensure that over 80% of the instances of that pattern in the corpus are for nouns, and likewise for verbs.

We test the model’s disambiguation in six conditions, with varying degrees of lexical bias. Unambiguous (‘noun/verb only’) conditions test words seen in the corpus only as nouns or verbs (10 words each). ‘Biased’ conditions test words with a clear bias (15 with average 93% noun bias; 15 with average 84% verb bias). An ‘equibias’ condition uses 4 words of approximately equal bias, and a novel word condition provides an unbiased case.

For the six sets of test words, we measure the effect of placing each of these words in both noun and verb contexts. That is, each word in each condition was used as the head word in each of the 500 noun and 500 verb disambiguating frames. For example, we create 500 frames where *book* is used as a noun, and 500 frames where it is used as a verb. We then use the fully-trained ‘combination’ model (Eq. (7)) to categorize each frame. Unlike in the previous experiment, we do not let the model create new clusters. For each frame, we choose the best-fitting existing cluster, then examine that cluster’s contents. As in experiment 2, we measure the proportions of each PoS of the frames in this cluster. We then average these measures over all tested frames in each condition.

6.2 Results

Figure 3 presents the measured PoS proportions for each of the six conditions. For both the equibias and

novel word conditions, we see that the clusters chosen for the noun context frames (labeled N) contain more nouns than verbs, and the clusters chosen for the verb context frames (V) contain more verbs than nouns. This suggests that although the model’s past experience with the head word is not sufficiently informative, the model can use the word’s context to disambiguate its category. In the ‘unambiguous’ and the ‘biased’ conditions, the head words’ lexical biases are too strong for the model to overcome.

However, the results show a realistic effect of the lexical bias. Note the contrasts from the ‘noun only’ condition, to the ‘noun biased’ condition, to ‘equibias’ (and likewise for the verb biases). As the lexical bias weakens, the counter-bias contexts (e.g., a noun bias with a verb context) show a stronger effect on the chosen clusters. This is a realistic effect of disambiguation seen in adults (MacDonald, 1993). Strongly biased words are more difficult to categorize in conflict with their bias than weakly biased words.

7 Related Work

Several existing computational models use distributional cues to find syntactic categories. Schütze (1993) employs co-occurrence statistics for common words, while Redington et al. (1998) build word distributional profiles using corpus bigram counts. Clark (2000) also builds distributional profiles, introducing an iterative clustering method to better handle ambiguity and rare words. Mintz (2003) shows that even very simple three-word templates can effectively define syntactic categories. Each of these models demonstrates that by using the kinds of simple information to which children are known to be sensitive, syntactic categories are learnable. However, the specific learning mechanisms they use, such as the hierarchical clustering methods of Redington et al. (1998), are not intended to be cognitively plausible.

In contrast, Cartwright and Brent (1997) propose

an incremental model of syntactic category acquisition that uses a series of linguistic preferences to find common patterns across sentence-length templates. Their model presents an important incremental algorithm which is very effective for discovering categories in artificial languages. However, the model's reliance on templates limits its applicability to transcripts of actual spoken language data, which contain high variability and noise.

Recent models that apply Bayesian approaches to PoS tagging are not incremental and assume a fixed number of tags (Goldwater and Griffiths, 2007; Toutanova and Johnson, 2008). In syntactic category acquisition, the true number of categories is unknown, and must be inferred from the input.

8 Conclusions and Future Directions

We have developed a computational model of syntactic category acquisition in children, and demonstrated its behaviour on a corpus of naturalistic child-directed data. The model is based on domain-general properties of feature similarity, in contrast to earlier, more linguistically-specific methods. The incremental nature of the algorithm contributes to a substantial improvement in psychological plausibility over previous models of syntactic category learning. Furthermore, due to its probabilistic framework, our model is robust to noise and variability in natural language.

Our model successfully uses a syntactic bootstrapping mechanism to build on the distributional properties of words. Using its existing partial knowledge of categories, the model applies a second level of analysis to learn patterns in the input. By making few assumptions about prior linguistic knowledge, the model develops realistic syntactic categories from the input data alone. The explicit bootstrapping component improves the model's ability to learn adult categories, and its learning trajectory resembles relevant behaviours seen in children. Using the contextual patterns of individual parts of speech, we show differential learning rates across nouns, verbs, and adjectives that mimic child development. We also show an effect of a lexical bias in category disambiguation.

The algorithm is currently only implemented as an incremental process. However, comparison with a batch version of the algorithm, such as by using a Gibbs sampler (Sanborn et al., 2006), would help us further understand the effect of incrementality on language fidelity.

While we have only examined the effects of learning categories from simple distributional information, the feature-based framework of our model could easily be extended to include other sources of information, such as morphological and phonological cues. Furthermore, it would also be possible to include se-

mantic features, thereby allowing the model to draw on correlations between semantic and syntactic categories in learning.

Acknowledgments

We thank Afra Alishahi for valuable discussions, and the anonymous reviewers for their comments. We gratefully acknowledge the financial support of NSERC of Canada and the University of Toronto.

References

- Alishahi, A. and S. Stevenson 2008. A computational model for early argument structure acquisition. *Cognitive Science*, 32(5).
- Anderson, J. R. 1991. The adaptive nature of human categorization. *Psychological Review*, 98(3):409–429.
- Cartwright, T. A. and M. R. Brent 1997. Syntactic categorization in early language acquisition: formalizing the role of distributional analysis. *Cognition*, 63:121–170.
- Clark, A. 2000. Inducing syntactic categories by context distribution clustering. In *CoNLL2000*, pp. 91–94.
- Goldwater, S. and T. L. Griffiths 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *Proc. of ACL2007*, pp. 744–751.
- Hubert, L. and P. Arabie 1985. Comparing partitions. *Journal of Classification*, 2:193–218.
- Kemp, N., E. Lieven, and M. Tomasello 2005. Young children's knowledge of the “determiner” and “adjective” categories. *J. Speech Lang. Hear. R.*, 48:592–609.
- MacDonald, M. C. 1993. The interaction of lexical and syntactic ambiguity. *J. Mem. Lang.*, 32:692–715.
- MacWhinney, B. 2000. *The CHILDES Project: Tools for analyzing talk*, volume 2: The Database. Lawrence Erlbaum, Mahwah, NJ, 3 edition.
- Mintz, T. H. 2003. Frequent frames as a cue for grammatical categories in child directed speech. *Cognition*, 90:91–117.
- Olguin, R. and M. Tomasello 1993. Twenty-five-month-old children do not have a grammatical category of verb. *Cognitive Development*, 8:245–272.
- Onnis, L. and M. H. Christiansen 2005. New beginnings and happy endings: psychological plausibility in computational models of language acquisition. *CogSci2005*.
- Redington, M., N. Chater, and S. Finch 1998. Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, 22(4):425–469.
- Sanborn, A. N., T. L. Griffiths, and D. J. Navarro 2006. A more rational model of categorization. *CogSci2006*.
- Schütze, H. 1993. Part of speech induction from scratch. In *Proc. of ACL1993*, pp. 251–258.
- Theakston, A. L., E. V. Lieven, J. M. Pine, and C. F. Rowland 2001. The role of performance limitations in the acquisition of verb-argument structure: an alternative account. *J. Child Lang.*, 28:127–152.
- Tomasello, M. 2000. Do young children have adult syntactic competence? *Cognition*, 74:209–253.
- Tomasello, M., N. Akhtar, K. Dodson, and L. Rekau 1997. Differential productivity in young children's use of nouns and verbs. *J. Child Lang.*, 24:373–387.
- Toutanova, K. and M. Johnson 2008. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *NIPS2008*.

Fully Unsupervised Graph-Based Discovery of General-Specific Noun Relationships from Web Corpora Frequency Counts

Gaël Dias
HULTIG
University of
Beira Interior
ddg@di.ubi.pt

Raycho Mukelov
HULTIG
University of
Beira Interior
raicho@hultig.di.ubi.pt

Guillaume Cleuziou
LIFO
University of
Orléans
cleuziou@univ-orleans.pt

Abstract.

In this paper, we propose a new methodology based on directed graphs and the TextRank algorithm to automatically induce general-specific noun relations from web corpora frequency counts. Different asymmetric association measures are implemented to build the graphs upon which the TextRank algorithm is applied and produces an ordered list of nouns from the most general to the most specific. Experiments are conducted based on the WordNet noun hierarchy and assess 65.69% of correct word ordering.

1 Introduction

Taxonomies are crucial for any knowledge-based system. They are in fact important because they allow to structure information, thus fostering their search and reuse. However, it is well known that any knowledge-based system suffers from the so-called knowledge acquisition bottleneck, i.e. the difficulty to actually model the domain in question. As stated in (Caraballo, 1999), WordNet has been an important lexical knowledge base, but it is insufficient for domain specific texts. So, many attempts have been made to automatically produce taxonomies (Grefenstette, 1994), but (Caraballo, 1999) is certainly the first work which proposes a complete overview of the problem by (1) automatically building a hierarchical structure of nouns based on bottom-up clustering methods and (2) labeling the internal nodes of the resulting tree with hypernyms from the nouns clustered underneath by using patterns such as “B is a kind of A”.

In this paper, we are interested in dealing with the second problem of the construction of an organized lexical resource i.e. discovering general-specific noun relationships, so that correct nouns are chosen to label internal nodes of any hierarchical knowledge base, such as the one proposed in (Dias et al., 2006). Most of the works proposed so far have (1) used predefined patterns or (2) automatically learned these patterns to identify hypernym/hyponym relationships. From the first paradigm, (Hearst, 1992) first identifies a set of lexico-syntactic patterns that are easily recognizable i.e. occur frequently and across text genre boundaries. These can be called seed patterns. Based on these seeds, she proposes a bootstrapping algorithm to semi-automatically acquire new more specific patterns. Similarly, (Caraballo, 1999) uses predefined patterns such as “X is a kind of Y” or “X, Y, and other Zs” to identify hypernym/hyponym relationships. This approach to information extraction is based on a technique called *selective concept extraction* as defined by (Riloff, 1993). Selective concept extraction is a form of text skimming that selectively processes relevant text while effectively ignoring surrounding text that is thought to be irrelevant to the domain.

A more challenging task is to automatically learn the relevant patterns for the hypernym/hyponym relationships. In the context of pattern extraction, there exist many approaches as summarized in (Stevenson and Greenwood, 2006). The most well-known work in this area is certainly the one proposed by (Snow et al., 2005) who use machine learning techniques to automatically replace hand-built knowledge. By using dependency path features extracted from parse trees, they introduce a general-purpose formalization and generalization of these patterns. Given a training set of text containing known hypernym pairs, their algorithm automatically extracts useful dependency paths and applies them to new corpora to identify novel pairs. (Sang and Hof-

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

mann, 2007) use a similar way as (Snow et al., 2006) to derive extraction patterns for hypernym/hyponym relationships by using web search engine counts from pairs of words encountered in WordNet. However, the most interesting work is certainly proposed by (Bollegala et al., 2007) who extract patterns in two steps. First, they find lexical relationships between synonym pairs based on snippets counts and apply wildcards to generalize the acquired knowledge. Then, they apply a SVM classifier to determine whether a new pair shows a relation of synonymy or not, based on a feature vector of lexical relationships. This technique could be applied to hypernym/hyponym relationships although the authors do not mention it.

On the one hand, links between words that result from manual or semi-automatic acquisition of relevant predicative or discursive patterns (Hearst, 1992; Carballo, 1999) are fine and accurate, but the acquisition of these patterns is a tedious task that requires substantial manual work. On the other hand, works done by (Snow et al., 2005; Snow et al., 2006; Sang and Hofmann, 2007; Bollegala et al., 2007) have proposed methodologies to automatically acquire these patterns mostly based on supervised learning to leverage manual work. However, training sets still need to be built.

Unlike other approaches, we propose an unsupervised methodology which aims at discovering general-specific noun relationships which can be assimilated to hypernym/hyponym relationships detection². The advantages of this approach are clear as it can be applied to any language or any domain without any previous knowledge, based on a simple assumption: specific words tend to attract general words with more strength than the opposite. As (Michelbacher et al., 2007) state: “there is a tendency for a strong forward association from a specific term like *adenocarcinoma* to the more general term *cancer*, whereas the association from *cancer* to *adenocarcinoma* is weak”. Based on this assumption, we propose a methodology based on directed graphs and the TextRank algorithm (Mihalcea and Tarau, 2004) to automatically induce general-specific noun relationships from web corpora frequency counts. Indeed, asymmetry in Natural Language Processing can be seen as a possible reason for

the degree of generality of terms (Michelbacher et al., 2007). So, different asymmetric association measures are implemented to build the graphs upon which the TextRank algorithm is applied and produces an ordered list of nouns, from the most general to the most specific. Experiments have been conducted based on the WordNet noun hierarchy and assessed that 65% of the words are ordered correctly.

2 Asymmetric Association Measures

In (Michelbacher et al., 2007), the authors clearly point at the importance of asymmetry in Natural Language Processing. In particular, we deeply believe that asymmetry is a key factor for discovering the degree of generality of terms. It is cognitively sensible to state that when someone hears about *mango*, he may induce the properties of a *fruit*. But, when hearing *fruit*, more common fruits will be likely to come into mind such as *apple* or *banana*. In this case, there exists an oriented association between *fruit* and *mango* (*mango* → *fruit*) which indicates that *mango* attracts more *fruit* than *fruit* attracts *mango*. As a consequence, *fruit* is more likely to be a more general term than *mango*.

Based on this assumption, asymmetric association measures are necessary to induce these associations. (Pecina and Schlesinger, 2006) and (Tan et al., 2004) propose exhaustive lists of association measures from which we present the asymmetric ones that will be used to measure the degree of attractiveness between two nouns, x and y , where $f(.,.)$, $P(.,.)$, $P(.,.)$ and N are respectively the frequency function, the marginal probability function, the joint probability function and the total of digrams.

$$\text{Braun - Blanquet} = \frac{f(x,y)}{\max(f(x,y)+f(x,\bar{y}), f(x,y)+f(\bar{x},y))} \quad (1)$$

$$\text{J measure} = \max \left[\begin{array}{l} P(x,y) \log \frac{P(y|x)}{P(y)} + P(x,\bar{y}) \log \frac{P(\bar{y}|\bar{x})}{P(\bar{y})} \\ P(x,y) \log \frac{P(x|y)}{P(x)} + P(\bar{x},y) \log \frac{P(x|y)}{P(x)} \end{array} \right] \quad (2)$$

$$\text{Confidence} = \max[P(x|y), P(y|x)] \quad (3)$$

$$\text{Laplace} = \max \left[\frac{N \cdot P(x,y) + 1}{N \cdot P(x) + 2}, \frac{N \cdot P(x,y) + 1}{N \cdot P(y) + 2} \right] \quad (4)$$

$$\text{Conviction} = \max \left[\frac{P(x) \cdot P(\bar{y})}{P(x,\bar{y})}, \frac{P(\bar{x}) \cdot P(y)}{P(\bar{x},y)} \right] \quad (5)$$

² We must admit that other kinds of relationships may be covered. For that reason, we will speak about general-specific relationships instead of hypernym/hyponym relationships.

$$\text{Certainty Factor} = \max \left[\frac{P(y|x) - P(y)}{1 - P(y)}, \frac{P(x|y) - P(x)}{1 - P(x)} \right] \quad (6)$$

$$\text{Added Value} = \max [P(y|x) - P(y), P(x|y) - P(x)] \quad (7)$$

All seven definitions show their asymmetry by evaluating the maximum value between two hypotheses i.e. by evaluating the attraction of x upon y but also the attraction of y upon x . As a consequence, the maximum value will decide the direction of the general-specific association i.e. ($x \rightarrow y$) or ($y \rightarrow x$).

3 TextRank Algorithm

Graph-based ranking algorithms are essentially a way of deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph. The basic idea implemented by a graph-based ranking model is that of voting or recommendation. When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Moreover, the importance of the vertex casting the vote determines how important the vote itself is, and this information is also taken into account by the ranking model. Hence, the score associated with a vertex is determined based on the votes that are cast for it, and the score of the vertices casting these votes.

Our intuition of using graph-based ranking algorithms is that more general words will be more likely to have incoming associations as they will be associated to many specific words. On the opposite, specific words will have few incoming associations as they will not attract general words (see Figure 1). As a consequence, the voting paradigm of graph-based ranking algorithms should give more strength to general words than specific ones, i.e. a higher voting score.

For that purpose, we first need to build a directed graph. Informally, if x attracts more y than y attracts x , we will draw an edge between x and y as follows ($x \rightarrow y$) as we want to give more credits to general words. Formally, we can define a directed graph $G = (V, E)$ with the set of vertices V (in our case, a set of words) and a set of edges E where E is a subset of $V \times V$ (in our case, defined by the asymmetric association measure value between two words). In Figure 1, we show the directed graph obtained by using the set of words $V = \{isometry, rate\ of\ growth, growth\ rate, rate\}$ randomly extracted from WordNet where *rate of*

growth and *growth rate* are synonyms, *isometry* an hyponym of the previous set and *rate* an hypernym of the same set. The weights associated to the edges have been evaluated by the confidence association measure (Equation 3) based on web search engine counts³.

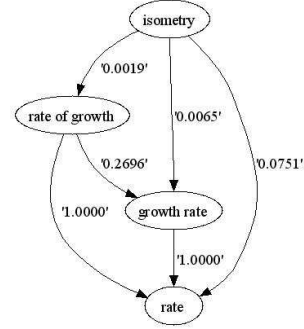


Fig. 1. Directed Graph based on synset #13153496 (*rate of growth, growth rate*) and its direct hypernym (*rate*) and hyponym (*isometry*).

Figure 1 clearly shows our assumption of generality of terms as the hypernym *rate* only has incoming edges whereas the hyponym *isometry* only has outgoing edges. As a consequence, by applying a graph-based ranking algorithm, we aim at producing an ordered list of words from the most general (with the highest value) to the most specific (with the lowest value). For that purpose, we present the TextRank algorithm proposed by (Mihalcea and Tarau, 2004) both for unweighted and weighted directed graphs.

3.1 Unweighted Directed Graph

For a given vertex V_i let $In(V_i)$ be the set of vertices that point to it, and let $Out(V_i)$ be the set of vertices that vertex V_i points to. The score of a vertex V_i is defined in Equation 8 where d is a damping factor that can be set between 0 and 1, which has the role of integrating into the model the probability of jumping from a given vertex to another random vertex in the graph⁴.

$$S(V_i) = (1 - d) + d \times \sum_{V_j \in In(V_i)} \frac{1}{|Out(V_j)|} \times S(V_j) \quad (8)$$

3.2 Weighted Directed Graph

In order to take into account the edge weights, a new formula is introduced in Equation 9.

³ We used counts returned by <http://www.yahoo.com>.

⁴ d is usually set to 0.85.

$$WS(V_i) = (1 - d) + d \times \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} \times WS(V_j) \quad (9)$$

After running the algorithm in both cases, a score is associated to each vertex, which represents the “importance” of the vertex within the graph. Notice that the final values obtained after TextRank runs to completion are not affected by the choice of the initial values randomly assigned to the vertices. Only the number of iterations needed for convergence may be different. As a consequence, after running the TextRank algorithm, in both its configurations, the output is an ordered list of words from the most general one to the most specific one. In table 1, we show both the lists with the weighted and unweighted versions of the TextRank based on the directed graph shown in Figure 1.

Unweighted		Weighted		WordNet	
$S(V_i)$	Word	$WS(V_i)$	Word	Categ.	Word
0.50	<i>rate</i>	0.81	<i>rate</i>	Hyper.	<i>rate</i>
0.27	<i>growth rate</i>	0.44	<i>growth rate</i>	Synset	<i>growth rate</i>
0.19	<i>rate of growth</i>	0.26	<i>rate of growth</i>	Synset	<i>rate of growth</i>
0.15	<i>isometry</i>	0.15	<i>isometry</i>	Hypo.	<i>isometry</i>

Table 1. TextRank ordered lists.

The results show that asymmetric measures combined with directed graphs and graph-based ranking algorithms such as the TextRank are likely to give a positive answer to our hypothesis about the degree of generality of terms. Moreover, we propose an unsupervised methodology for acquiring general-specific noun relationships. However, it is clear that deep evaluation is needed.

4 Experiments and Results

Evaluation is classically a difficult task in Natural Language Processing. In fact, as human evaluation is time-consuming and generally subjective even when strict guidelines are provided, measures to automatically evaluate experiments must be proposed. In this section, we propose three evaluation measures and discuss the respective results.

4.1 Constraints

WordNet can be defined as applying a set of constraints to words. Indeed, if word w is the hypernym of word x , we may represent this relation by the following constraint $y \succ x$, where \succ is the order operator stating that y is more general than x . As a consequence, for each set of three

synsets (the hypernym synset, the seed synset and the hyponym synset), a list of constraints can be established i.e. all words of the hypernym synset must be more general than all the words of the seed synset and the hyponym synset, and all the words of the seed synset must be more general than all the words in the hyponym synset. So, if we take the synsets presented in Table 1, we can define the following set of constraints: $\{rate \succ growth\ rate, rate \succ rate\ of\ growth, growth\ rate \succ isometry, rate\ of\ growth \succ isometry\}$.

In order to evaluate our list of words ranked by the level of generality against the WordNet categorization, we just need to measure the proportion of constraints which are respected as shown in Equation (10). We call, *correctness* this measure.

$$correctness = \frac{\# \text{ of common constraint}}{\# \text{ of constraint}} \quad (10)$$

For example, in Table 1, all the constraints are respected for both weighted and unweighted graphs, giving 100% correctness for the ordered lists compared to WordNet categorization.

4.2 Clustering

Another way to evaluate the quality of the ordering of words is to apply hard clustering to the words weighted by their level of generality. By evidencing the quality of the mapping between three hard clusters generated automatically and the hypernym synset, the seed synset and the hyponym synset, we are able to measure the quality of our ranking. As a consequence, we propose to (1) perform 3-means clustering over the list of ranked words, (2) classify the clusters by level of generality and (3) measure the precision, recall and f-measure of each cluster sorted by level of generality with the hypernym synset, the seed synset and the hyponym synset.

For the first task, we use the implementation of the k-means algorithm of the NLTK toolkit⁵. In particular, we bootstrap the k-means by choosing the initial means as follows. For the first mean, we choose the weight (the score) of the first word in the TextRank generated list of words. For the second mean, we take the weight of the middle word in the list and for the third mean, the weight of the last word in the list.

For the second task the level of generality of each cluster is evaluated by the average level of

⁵ <http://nltk.sourceforge.net/>

generality of words inside the cluster (or said with other words by its mean).

For the third task, the most general cluster and the hypernym synset are compared in terms of precision, recall and f-measure as shown in Equation (11), (12) and (13)⁶. The same process is applied to the second most general cluster and the seed synset, and the third cluster and the hyponym synset.

$$precision = \frac{Cluster \cap Synset}{|Cluster|} \quad (11)$$

$$recall = \frac{Cluster \cap Synset}{|Synset|} \quad (12)$$

$$f - measure = \frac{2 \times recall \times precision}{precision + recall} \quad (13)$$

4.3 Rank Coefficient Test

The evaluation can be seen as a rank test between two ordered lists. Indeed, one way to evaluate the results is to compare the list of general-specific relationships encountered by the TextRank algorithm and the original list given by WordNet. However, we face one problem. WordNet does not give an order of generality inside synsets. In order to avoid this problem, we can order words in each synset by their estimated frequency given by WordNet⁷ as well as their frequency calculated by web search hits. An example of both ordered lists is given in Table 2 for the synset #6655336 and its immediate hypernyms and hyponyms.

WordNet Estimated Frequency		Web Estimated Frequency	
Category	Word	Category	Word
Hypernym	<i>statement</i>	Hypernym	<i>statement</i>
Synset	<i>answer</i>	Synset	<i>reply</i>
Synset	<i>reply</i>	Synset	<i>response</i>
Synset	<i>response</i>	Synset	<i>answer</i>
Hyponym	<i>rescript</i>	Hyponym	<i>feedback</i>
Hyponym	<i>feedback</i>	Hyponym	<i>rescript</i>

Table 2. Estimated Frequency ordered lists for synset #6655336.

For that purpose, we propose to use the Spearman’s rank correlation coefficient (Rho). The Spearman’s Rho is a statistical coefficient that shows how much two random variables are cor-

related. It is defined in Equation (14) where d is the distance between every pair of words in the list ordered with TextRank and the reference list which is ordered according to WordNet or the Web and n is the number of pairs of ranked words.

$$\rho = 1 - \frac{6 \times \sum d_i^2}{n(n^2 - 1)} \quad (14)$$

In particular, the Spearman’s rank correlation coefficient is a number between -1 (no correlation at all) and 1 (very strong correlation).

4.4 Experiments

In order to evaluate our methodology, we randomly⁸ extracted 800 seed synsets for which we retrieved their hypernym and hyponym synsets. For each seed synset, we then built the associated directed weighted and unweighted graphs based on the asymmetric association measures referred to in section 2⁹ and ran the TextRank algorithm to produce a general-specific ordered lists of terms.

4.4.1 Results by Constraints

In Table 3, we present the results of the correctness for all seven asymmetric measures, both for the unweighted and weighted graphs.

Equation	Type of Graph	Correctness
Braun-Blanquet	Unweighted	65.68%
	Weighted	65.52%
J measure	Unweighted	60.00%
	Weighted	60.34%
Confidence	Unweighted	65.69%
	Weighted	65.40%
Laplace	Unweighted	65.69%
	Weighted	65.69%
Conviction	Unweighted	61.81%
	Weighted	63.39%
Certainty Factor	Unweighted	65.59%
	Weighted	63.76%
Added Value	Unweighted	65.61%
	Weighted	64.90%
Baseline ¹⁰	None	55.68%

Table 3. Results for the Evaluation by Constraints.

The best results are obtained by the Confidence and the Laplace measures reaching 65.69% cor-

⁶ Where $Cluster \cap Synset$ means the number of words common to both Synset and Cluster, and $|Synset|$ and $|Cluster|$ respectively measure the number of words in the Synset and the Cluster.

⁷ We use WordNet 2.1.

⁸ We guarantee 98% significance level for an error of 0.05 following the normal distribution.

⁹ The probability functions are estimated by the Maximum Likelihood Estimation (MLE).

¹⁰ The baseline is the list of words ordered by web hits frequency (without TextRank).

rectness. However, the Braun-Blanquet, the Certainty Factor and the Added Value give results near the best ones. Only the J measure and the Conviction metric seem to perform worst.

It is also important to note that the difference between unweighted and weighted graphs is marginal which clearly points at the fact that the topology of the graph is more important than its weighting. This is also confirmed by the fact that most of the asymmetric measures perform alike.

4.4.2 Results by Clustering

In Table 4, we present the results of precision, recall and f-measure for both weighted and unweighted graphs for all the seven asymmetric measures. The best precision is obtained for the weighted graph with the Confidence measure evidencing 47.62% and the best recall is also obtained by the Confidence measure also for the weighted graph reaching 47.68%. Once again, the J measure and the Conviction metric perform worst showing worst f-measures. Contrarily, the Confidence measure shows the best performance in terms of f-measure for the weighted graph, i.e. 47.65% while the best result for the unweighted graphs is obtained by the Certainty factor with 46.50%.

These results also show that the weighting of the graph plays an important issue in our methodology. Indeed, most metrics perform better with weighted graphs in terms of f-measure.

Equation	Graph	Precision	Recall	F-measure
Braun-Blanquet	Unweighted	46.61	46.06	46.33
	Weighted	47.60	47.67	47.64
J measure	Unweighted	40.92	40.86	40.89
	Weighted	42.61	43.71	43.15
Confidence	Unweighted	46.54	46.02	46.28
	Weighted	47.62	47.68	47.65
Laplace	Unweighted	46.67	46.11	46.39
	Weighted	46.67	46.11	46.39
Conviction	Unweighted	42.13	41.67	41.90
	Weighted	43.62	43.99	43.80
Certainty Factor	Unweighted	46.49	46.52	46.50
	Weighted	44.84	45.85	45.34
Added Value	Unweighted	46.61	46.59	46.60
	Weighted	47.13	47.27	47.19

Table 4. Results for the Evaluation by Clustering.

In Table 5, 6 and 7, we present the same results as in Table 4 but at different levels of analysis i.e. precision, recall and f-measure at hypernym, seed and hyponym levels. Indeed, it is important to understand how the methodology performs at different levels of generality as we verified that

our approach performs better at higher levels of generality.

Equation	Graph	Precision	Recall	F-measure
Braun-Blanquet	Unweighted	59.38	37.38	45.88
	Weighted	58.75	39.35	47.14
J measure	Unweighted	46.49	37.00	41.20
	Weighted	47.19	41.90	44.38
Confidence	Unweighted	59.20	37.30	45.77
	Weighted	58.71	39.22	47.03
Laplace	Unweighted	59.50	37.78	45.96
	Weighted	59.50	37.78	45.96
Conviction	Unweighted	50.07	35.88	41.80
	Weighted	52.72	40.74	45.96
Certainty Factor	Unweighted	55.90	38.29	45.45
	Weighted	51.64	42.93	46.88
Added Value	Unweighted	56.26	37.90	45.29
	Weighted	58.21	40.09	47.48

Table 5. Results at the hypernym level.

Equation	Graph	Precision	Recall	F-measure
Braun-Blanquet	Unweighted	43.05	37.86	40.29
	Weighted	46.38	33.14	38.66
J measure	Unweighted	40.82	43.72	42.22
	Weighted	43.98	33.89	38.28
Confidence	Unweighted	43.03	37.67	40.17
	Weighted	46.36	33.02	38.57
Laplace	Unweighted	43.10	37.78	40.27
	Weighted	43.10	37.78	40.27
Conviction	Unweighted	40.36	38.02	39.16
	Weighted	42.60	26.39	32.59
Certainty Factor	Unweighted	44.28	40.87	42.51
	Weighted	44.14	40.70	42.35
Added Value	Unweighted	44.21	40.74	42.40
	Weighted	45.78	32.90	38.29

Table 6. Results at the seed level.

Equation	Graph	Precision	Recall	F-measure
Braun-Blanquet	Unweighted	37.39	62.96	46.92
	Weighted	37.68	70.50	49.12
J measure	Unweighted	35.43	41.87	38.38
	Weighted	36.69	55.33	44.12
Confidence	Unweighted	37.38	63.09	46.95
	Weighted	37.79	70.80	49.27
Laplace	Unweighted	37.40	63.11	46.97
	Weighted	37.40	63.11	46.97
Conviction	Unweighted	35.97	50.94	42.16
	Weighted	35.54	64.85	45.92
Certainty Factor	Unweighted	39.28	60.40	47.60
	Weighted	38.74	53.92	45.09
Added Value	Unweighted	39.36	61.15	47.89
	Weighted	37.39	68.81	48.45

Table 7. Results at the hyponym level.

Indeed, the precision scores go down from 59.50% at the hypernym level to 39.36% at the hyponym level with 46.38% at the seed level. The same phenomenon is inversely true for the recall with 42.93% at the hypernym level,

43.72% at the seed level and 70.80% at the hyponym level.

This situation can easily be understood as most of the clusters created by the k-means present the same characteristics i.e. the upper level cluster usually has fewer words than the middle level cluster which in turn has fewer words than the last level cluster. As a consequence, the recall is artificially high for the hyponym level. But on the opposite, the precision is high for higher levels of generality which is promising for the automatic construction of hierarchical thesauri. Indeed, our approach can be computed recursively so that each level of analysis is evaluated as if it was at the hypernym level, thus taking advantage of the good performance of our approach at upper levels of generality¹¹.

4.4.3 Results by Rank Test

For each produced list, we calculated the Spearman's Rho both with WordNet and Web Estimated Lists for weighted and unweighted graphs. Table 8 presents the average results for the 800 randomly selected synsets.

Equation	Type of Graph	Rho with WNet Est. list	Rho with Web Est. list
Braun-Blanquet	Unweighted	0.38	0.30
	Weighted	0.39	0.39
J measure	Unweighted	0.23	0.19
	Weighted	0.27	0.27
Confidence	Unweighted	0.38	0.30
	Weighted	0.39	0.39
Laplace	Unweighted	0.38	0.30
	Weighted	0.38	0.38
Conviction	Unweighted	0.30	0.22
	Weighted	0.33	0.33
Certainty Factor	Unweighted	0.38	0.29
	Weighted	0.35	0.35
Added Value	Unweighted	0.37	0.29
	Weighted	0.38	0.38
Baseline ¹²	None	0.14	0.14

Table 8. Results for the Spearman's rank correlation coefficient.

Similarly to what we evidenced in section 4.4.1., the J measure and the Conviction metric are the measures which less seem to map the correct order by evidencing low correlation scores. On the other hand, the Confidence metric still gives the best results equally with the Laplace and Braun-Blanquet metrics.

¹¹ This will be studied as future work.

¹² The baseline is the list of words ordered by web hits frequency.

It is interesting to note that in the case of the web estimated list, the weighted graphs evidence much better results than the unweighted ones, although they do not show improved results compared to the WordNet list. On the one hand, these results show that our methodology is capable to map to WordNet lists as easily as to Web lists even that it is based on web frequency counts. On the other hand, the fact that weighted graphs perform best, shows that the topology of the graph lacks in accuracy and needs the application of weights to counterpoint this lack.

4.5 Discussion

An important remark needs to be made at this point of our explanation. There is a large ambiguity introduced in the methodology by just looking at web counts. Indeed, when counting the occurrences of a word like *answer*, we count all its occurrences for all its meanings and forms. For example, based on WordNet, the word *answer* can be a verb with ten meanings and a noun with five meanings. Moreover, words are more frequent than others although they are not so general, unconfirming our original hypothesis. Looking at Table 2, *feedback* is a clear example of this statement. As we are not dealing with a single domain within which one can expect to see the "one sense per discourse" paradigm, it is clear that the Rho coefficient would not be as good as expected as it is clearly biased by "incorrect" counts. One direct implication of this comment is the use of web estimated lists to evaluate the methodology.

Also, there has been a great discussion over the last few months in the corpora list¹³ whether one should use web counts instead of corpus counts to estimate word frequencies. In our study, we clearly see that web counts show evident problems, like the ones mentioned by (Kilgarriff, 2007). However, they cannot be discarded so easily. In particular, we aim at looking at web counts in web directories that would act as specific domains and would reduce the space for ambiguity. Of course, experiments with well-known corpora will also have to be made to understand better this phenomenon.

5 Conclusions and Future Work

In this paper, we proposed a new methodology based on directed weighted/unweighted graphs and the TextRank algorithm to automatically in-

¹³ Finalized by (Kilgarriff, 2007).

duce general-specific noun relationships from web corpora frequency counts. To our knowledge, such an unsupervised experiment has never been attempted so far. In order to evaluate our results, we proposed three different evaluation metrics. The results obtained by using seven asymmetric association measures based on web frequency counts showed promising results reaching levels of (1) constraint coherence of 65.69%, (2) clustering mapping of 59.50% in terms of precision for the hypernym level and 42.72% on average in terms of f-measure and (3) ranking similarity of 0.39 for the Spearman's rank correlation coefficient.

As future work, we intend to take advantage of the good performance of our approach at the hypernym level to propose a recursive process to improve precision results over all levels of generality.

Finally, it is important to notice that the evaluation by clustering evidences more than a simple evaluation of the word order, but shows how this approach is capable to automatically map clusters to WordNet classification.

References

- Bollegala, D., Matsuo, Y. and Ishizuka, M. 2007. *Measuring Semantic Similarity between Words Using WebSearch Engines*. In Proceedings of International World Wide Web Conference (WWW 2007).
- Caraballo, S.A. 1999. *Automatic Construction of a Hypernym-labeled Noun Hierarchy from Text*. In Proceedings of the Conference of the Association for Computational Linguistics (ACL 1999).
- Dias, G., Santos, C., and Cleuziou, G. 2006. *Automatic Knowledge Representation using a Graph-based Algorithm for Language-Independent Lexical Chaining*. In Proceedings of the Workshop on Information Extraction Beyond the Document associated to the Joint Conference of the International Committee of Computational Linguistics and the Association for Computational Linguistics (COLING/ACL), pages. 36-47.
- Grefenstette, G. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, USA.
- Hearst, M.H. 1992. *Automatic Acquisition of Hyponyms from Large Text Corpora*. In Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING 1992), pages 539-545.
- Kilgarriff, A. 2007. Googleology is Bad Science. *Computational Linguistics* 33 (1), pages: 147-151.
- Michelbacher, L., Evert, S. and Schütze, H. 2007. *Asymmetric Association Measures*. In Proceedings of the Recent Advances in Natural Language Processing (RANLP 2007).
- Mihalcea, R. and Tarau, P. 2004. *TextRank: Bringing Order into Texts*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), pages 404-411.
- Pecina, P. and Schlesinger, P. 2006. *Combining Association Measures for Collocation Extraction*. In Proceedings of the International Committee of Computational Linguistics and the Association for Computational Linguistics (COLING/ACL 2006).
- Riloff, E. 1993. *Automatically Constructing a Dictionary for Information Extraction Tasks*. In Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI 1993), pages 811-816.
- Sang, E.J.K. and Hofmann, K. 2007. *Automatic Extraction of Dutch Hypernym-Hyponym Pairs*. In Proceedings of Computational Linguistics in the Netherlands Conference (CLIN 2007).
- Snow, R., Jurafsky, D. and Ng, A. Y. 2005. *Learning Syntactic Patterns for Automatic Hypernym Discovery*. In Proceedings of the International Committee of Computational Linguistics and the Association for Computational Linguistics (COLING/ACL 2006).
- Snow, R., Jurafsky, D. and Ng, A. Y. 2005. *Semantic Taxonomy Induction from Heterogenous Evidence*. In Proceedings of the Neural Information Processing Systems Conference (NIPS 2005).
- Stevenson, M., and Greenwood, M. 2006. *Comparing Information Extraction Pattern Models*. In Proceedings of the Workshop on Information Extraction Beyond the Document associated to the Joint Conference of the International Committee of Computational Linguistics and the Association for Computational Linguistics (COLING/ACL 2006), pages. 29-35.
- Tan, P.-N., Kumar, V. and Srivastava, J. 2004. *Selecting the Right Objective Measure for Association Analysis*. *Information Systems*, 29(4). pages 293-313.

Acquiring Knowledge from the Web to be used as Selectors for Noun Sense Disambiguation

Hansen A. Schwartz and Fernando Gomez

School of Electrical Engineering and Computer Science

University of Central Florida

{hschwartz, gomez}@cs.ucf.edu

Abstract

This paper presents a method of acquiring knowledge from the Web for noun sense disambiguation. Words, called selectors, are acquired which take the place of an instance of a target word in its local context. The selectors serve for the system to essentially learn the areas or concepts of WordNet that the sense of a target word should be a part of. The correct sense is chosen based on a combination of the strength given from similarity and relatedness measures over WordNet and the probability of a selector occurring within the local context. Our method is evaluated using the coarse-grained all-words task from SemEval 2007. Experiments reveal that path-based similarity measures perform just as well as information content similarity measures within our system. Overall, the results show our system is out-performed only by systems utilizing training data or substantially more annotated data.

1 Introduction

Recently, the Web has become the focus for many word sense disambiguation (WSD) systems. Due to the limited amount of sense tagged data available for supervised approaches, systems which are typically referred to as unsupervised, have turned to the use of unannotated corpora including the Web. The advantage of these systems is that they can disambiguate all words, and not just a set of words for which training data has been provided. In this paper we present an unsupervised system which uses the Web in a novel fashion to perform

sense disambiguation of any noun, incorporating both similarity and relatedness measures.

As explained in (Brody et al., 2006), there are generally two approaches to unsupervised WSD. The first is referred to as *token* based, which compares the relatedness of a target word to other words in its context. The second approach is *type* based, which uses or identifies the most common sense of a word over a discourse or corpus, and annotates all instances of a word with the most common sense. Although the *type* based approach is clearly bound to fail occasionally, it is commonly found to produce the strongest results, rivaling supervised systems (McCarthy et al., 2004). We identify a third approach through the use of *selectors*, first introduced by (Lin, 1997), which help to disambiguate a word by comparing it to other words that may replace it within the same local context.

We approach the problem of word sense disambiguation through a relatively straightforward method that incorporates ideas from the *token*, *type*, and *selector* approaches. In particular, we expand the use of *selectors* in several ways. First, we revise the method for acquiring selectors to be applicable to the web, a corpus that is, practically speaking, impossible to parse in whole. Second, we describe a path-based similarity measure that is more suited for a portion of our method than the relatedness measures used by *token* based systems. Finally, we expand the use of selectors to help with disambiguating nouns other than the one replaced.

2 Background

2.1 Word Sense Disambiguation

A popular approach to using the web or unannotated corpora for word sense disambiguation involves the use of monosemous relatives. Monosemous relatives are words which are similar to a

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

sense of the target word, but which only have one sense. By searching text for these words, one can build training data for each sense of a target word. This idea was proposed by (Leacock et al., 1998). More recently, the idea has been used to automatically create sense tagged corpora (Mihalcea, 2002; Agirre and Martinez, 2004). These methods queried large corpora with relatives rather than with the context.

With some resemblances to our approach, (Martinez et al., 2006) present the *relatives in context* method. A key similarity of this method with ours is the use of context in the web queries. They produce queries with relatives in place of the target word in a context with a window size of up to 6. Similarly, (Yuret, 2007) first chooses substitutes and determines a sense by looking at the probability of a substitute taking the place of the target word within the Web1T corpus. The number of hits each query has on the web is then used to pick the correct sense. Our approach differs from these in that we acquire words(selectors) from the web, and proceed to choose a sense based on similarity measures over WordNet (Miller et al., 1993). We also attempt to match the context of the entire sentence if possible, and we are more likely to receive results from longer queries by including the wildcard instead of pre-chosen relatives.

We adopted the term *selector* from (Lin, 1997) to refer to a word which takes the place of another in the same local context. Lin searched a local context database, created from dependency relationships over an unannotated corpora in order to find selectors. In this case, the local context was represented by the dependency relationships. Given that the task of producing a dependency parse database of the Web is beyond our abilities, we search for the surrounding local context as text in order to retrieve selectors for a given word. Another difference is that we compare the relatedness of selectors of other words in the sentence to the target word, and we also incorporate a path-based similarity measure along with a gloss-based relatedness measure.

2.2 Similarity and Relatedness Measures

Semantic similarity and relatedness measures have an extensive history. The measures reported in this work were included based on appropriateness with our approach and because of past success according to various evaluations (Patwardhan et al., 2003;

Budanitsky and Hirst, 2006).

Many similarity measures have been created which only use paths in the WordNet ontology. One approach is to simply compute the length of the shortest path between two concepts over the hypernym/hyponym relationship (Rada et al., 1989). Other methods attempt to compensate for the uniformity problem, the idea that some areas of the ontology are more dense than others, and thus all edges are not equal. (Wu and Palmer, 1994) uses the path length from the root to the lowest common subsumer(LCS) of two concepts scaled by the distance from the LCS to each concept. Another method, by (Leacock et al., 1998), normalizes path distance based on the depth of hierarchy. Our method attempts to produce a normalized depth based on the average depth of all concepts which are leaf nodes below the lowest common subsumer in a tree.

We employ several other measures in our system. These measures implement various ideas such as *information content* (Jiang and Conrath, 1997; Lin, 1997) and *gloss overlaps* (Banerjee and Pedersen, 2003). For our work the path-based and information content measures are referred to as *similarity measures*, while the gloss-based methods are referred to as *relatedness measures*. Relatedness measures can be used to compare words from different parts of speech. In past evaluations of token based WSD systems, information content and gloss-based measures perform better than path-based measures (Patwardhan et al., 2003; Budanitsky and Hirst, 2006).

3 Method

The general idea of our method is to find the sense of a target noun which is most similar to all selectors which can replace the target and most related to other words in context and their selectors. Our method requires that a test sentence has been part-of-speech tagged with noun, verb, and adjective POS, and we use the selectors from all of these parts of speech as well as noun selectors of pronouns and proper nouns. In this work, we only disambiguate nouns because *similarity* measures for target selectors are based heavily on the depth that is present in the WordNet noun ontology. However, we are still able to use verb and adjective selectors from the context through *relatedness* measures working over all parts of speech listed. The method can be broken into two steps:

1. Acquire probabilities of selectors occurring for all nouns, verbs, adjectives, pronouns and proper nouns from the Web.
2. Rank the senses of a target noun according to similarity with its own selectors and relatedness with other selectors in the context.

These steps are described in detail below. Finally, we also describe a similarity measure we employ.

3.1 Acquiring Selectors

We acquire *target selectors* and *context selectors* from the Web. Target selectors are those words which replace the current target word in the local context, while *context selectors* are words which may replace other words in the local context. There are four different types of context selectors:

noun context selectors essentially the target selectors for other nouns of the sentence.

verb context selectors verbs which are found to replace other verbs in the sentence.

adjective context selectors adjectives which replace other adjectives in the sentence.

pro context selectors nouns which replace pronouns and proper nouns.

A query must be created based on the original sentence and target word. This is fairly straightforward as the target word is removed and replaced with a * to indicate the wildcard. For example, when searching for selectors of “batter” from “She put the batter in the refrigerator.”, a query of “She put the * in the refrigerator.” is used. The queries are sent through the Yahoo! Search Web Services¹ in order to retrieve matching text on the web.

The selectors are extracted from the samples returned from the web by matching the wildcard of the query to the sample. The wildcard match is thrown out if any of the following conditions are true: *longer than 4 words, contains any punctuation, is composed only of pronouns or the original word*. Keep in mind we acquire the nouns that replace the pronouns of the original sentence, so a selector is never a pronoun. WordNet is used to determine if the phrase is a compound and the base morphological form of the head word. Results containing head words not found in WordNet are filtered out. Proper nouns are used if they are found in WordNet. Finally, the list of selectors is

adjusted so no single word takes up more than 30% of the list.

The Web is massive, but unfortunately it is not large enough to find results when querying with a whole sentence a majority of the time. Therefore, we perform truncation of the query to acquire more selectors. For this first work with selectors from the web, we chose to create a simple truncation focused just on syntax in order to run quickly. The steps below are followed and the final step is repeated until a stop condition is met.

- i Shorten to a size of 10 words.
- ii Remove end punctuation, if not preceded by *.
- iii Remove front punctuation, if not preceded by *.
- iv Remove determiners (*the, a, an, this, that*) preceding *.
- v Remove a single word.

When removing a single word, the algorithm attempts to keep the * in the center. Figure 1 demonstrates the loop that occurs until a stop condition is met: enough selectors are found or the query has reached a minimum size. Since a shorter query should return the same results as a longer query, we filter the selectors from longer query results out of the shorter results. It is important that the criteria to continue searching is based on the number of selectors and not on the number of samples, because many samples fail to produce a selector. Validation experiments were performed to verify that each step of truncation was helpful in returning more results with valid selectors, although the results are not reported as the focus is on the method in general. Selectors are tied to the queries used to acquire them in order to help emphasize results from longer queries.

The steps to acquire all types of selectors (target or any in context) are the same. The part of speech only plays a part in determining the base form or compounds when using WordNet. Note that all selectors for each noun, verb, adjective, and pronoun/proper can be acquired in one pass, so that duplicate queries are not sent to the Web. When the process is complete we have a probability value for each selector word (w_s) to occur in a local context given by the acquisition query (q). The probability of w_s appearing in q is denoted as:

$$p_{occ}(w_s, q)$$

3.2 Ranking Senses

There are essentially two assumptions made in order to rank the senses of a noun.

¹<http://developer.yahoo.com/search/>

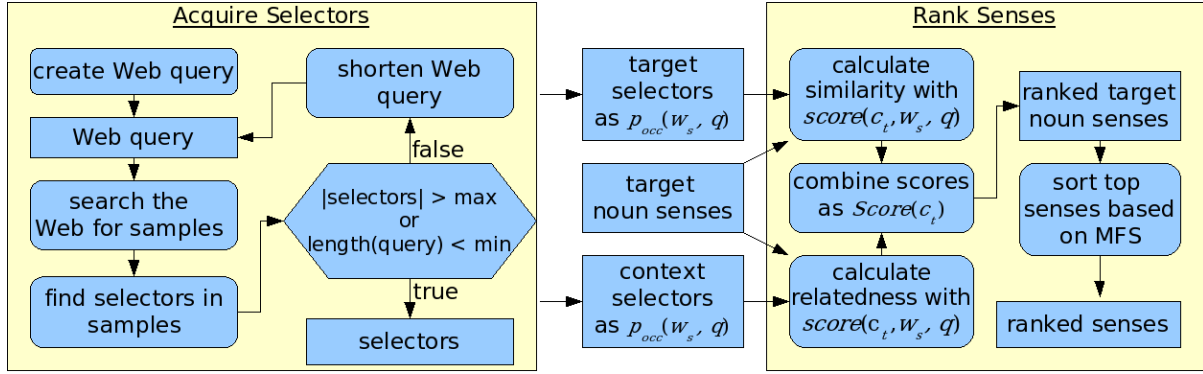


Figure 1: The overall process undertaken to disambiguate a noun. (Note that selectors only need to be acquired once for each sentence since they can be reused for each target noun.)

1. Similar concepts (or noun senses) appear in similar syntactic constructions.
2. The meaning of a word is often related to other words in its context

The first assumption implies the use of a similarity measure with *target selectors*. The meaning of the target selectors should be very similar to that of the original word, and thus we compare similarity between all target selectors with each sense of the original word.

The second assumption reflects the information provided by *context selectors*, for which we use a relatedness measure to compare with the original word. Note that because context selectors may be of a different part of speech, we should be sure this measure is able to handle multiple parts of speech.

Regardless of the similarity or relatedness measure used, the value produced is applied the same for both *target selectors* and *context selectors*. We are comparing the senses (or concepts) of the original target word with all of the selectors. To find the similarity or relatedness of two words, rather than two concepts, one can use the maximum value over all concepts of the selector word and all the senses of the target word, (Resnik, 1999, *word similarity*):

$$wsr(w_t, w_s) = \max_{c_t, c_s} [srm(c_t, c_s)]$$

where *srm* is a similarity or relatedness measure and c_t, c_s represent a sense (concept) of the target word (w_t) and selector word (w_s) respectively. We would like to get a value for each sense of a target word if possible, so we derive similarity or relatedness between one concept and one word as:

$$cwsr(c_t, w_s) = \max_{c_s} [srm(c_t, c_s)]$$

Intuitively, combining *cwsr* with p_{occ} is the basis for scoring the senses of each noun. However, we also take several other values into account, in order to learn most effectively from Web selectors. The score is scaled by the number of senses of the selector and the length of the query used to acquire it. This gives less ambiguous selectors and those selectors with a most similar local context a stronger role. These values are represented by $senses(w_s)$ and $qweight = \frac{current_length}{original_length}$:

$$\begin{aligned} score(c_t, w_s, q) \\ = p_{occ}(w_s, q) * cwsr(c_t, w_s) * \frac{qweight}{senses(w_s)} \end{aligned}$$

The scores are summed with:

$$sum_{type}(c_t) = \sum_q \sum_{w_s} score(c_t, w_s, q)$$

where q ranges over all queries for a type (*type*) of selector, and w_s ranges over all selectors acquired with query q .

Overall, the algorithm gives a score to each sense by combining the normalized sums from all types of the selectors:

$$Score(c_t) = \sum_{type} \frac{sum_{type}(c_t)}{\max_{c \in w_t} [sum_{type}(c)]} * scale_{type}$$

where *typ* ranges over a type of selector (target, noun context, verb context, adjective context, pro context), c ranges over all senses of the target word (w_t), and $scale_{type}$ is a constant for each type of selector. We experimented with different values over 60 instances of the corpus to decide on a scale value of 1 for *target selectors*, a value of 0.5 for

noun and verb *context selectors*, and a value of 0.1 for adjective and pro *context selectors*. This weights the scores that come from target selectors equal to that of noun and verb context selectors, while the adjective and pro selectors only play a small part.

Finally, the senses are sorted based on their *Score*, and we implement the most frequent sense heuristic as a backoff strategy. All those senses within 5% of the top sense’s *Score*, are re-sorted, ranking those with lower sense numbers in WordNet higher. The highest ranking sense is taken to be the predicted sense.

3.3 Similarity Measure

We use the notion that similarity is a specific type of relatedness (Rada et al., 1989; Patwardhan et al., 2003). For our purposes, a *similarity* measure is used for nouns which may take the place of a target word within its local context, while words which commonly appear in other parts of the local context are measured by *relatedness*. In particular, the *similarity* measure places emphasis strictly on the *is-a* relationship. As an example, “bottle” and “water” are *related* but not *similar*, while “cup” and “bottle” are *similar*. Because of this distinction, we would classify our path-based measure as a *similarity* measure.

A well known problem with path-based measures is the assumption that the links between concepts are all uniform (Resnik, 1999). As a response to this problem, approaches based on information content are used, such as (Resnik, 1999; Jiang and Conrath, 1997; Lin, 1997). These measures still use the *is-a* relationship in WordNet, but they do not rely directly on edges to determine the strength of a relationship between concepts. (Patwardhan et al., 2003) shows that measures based on information content or even gloss based measures generally perform best for comparing a word with other words in its context for word sense disambiguation. However, these measures may not be as suited for relating one word to other words which may replace it (*target selectors*). Therefore, our similarity measure examines the use of links in WordNet, and attempts to deal with the uniformity problem by normalizing depths based on average leaf node depth.

All types of relatedness measures return a value representing the strength of the relation between the two concepts. These values usually range be-

tween 0 and 1. Note that concepts are not the same as words, and the example above assumes one chooses the sense of “water” as a liquid and the sense of “bottle” and “cup” as a container. Our similarity measure is based on finding the normalized depth (*nd*) of a concept (*c*) in the WordNet Hierarchy:

$$nd(c) = \frac{depth(c)}{ald(c)}$$

Where *depth* is the length from the concept to the root, and *ald* returns the average depth of all descendants (hyponyms) that do not have hyponyms themselves (average leaf depth):

$$ald(c) = \frac{\sum_{L \in lnodes(c)} depth(l)}{|lnodes(c)|}$$

To be clear, *lnodes* returns a list of only those nodes without hyponyms that are themselves hyponyms of *c*. We chose to only use the leaf depth as opposed to all depths of descendants, because *ald* produces a value representing maximum depth for that branch in the tree, which is more appropriate for normalization.

Like other similarity measures, for any two concepts we compute the lowest (or deepest) common subsumer, *lcs*, which is the deepest node in the hierarchy which is a hypernym of both concepts. The similarity between two concepts is then given by the normalized depth of their *lcs*:

$$sim(c_1, c_2) = nd(lcs(c_1, c_2))$$

Thus, a concept compared to itself will have a score of 1, while the most dissimilar concepts will have a score of 0. Following (Wu and Palmer, 1994; Lin, 1997) we scale the measure by each concept’s *nd* as follows:

$$scaled_sim(c_1, c_2) = \frac{2 * sim(c_1, c_2)}{nd(c_1) + nd(c_2)}$$

where our *normalized depth* replaces the *depth* or *information content* value used by the past work.

4 Evaluation

We evaluated our algorithm using the SemEval 2007 coarse-grained all-words task. In order to achieve a coarse grained sense inventory WordNet 2.1 senses were manually mapped to the top-level of the Oxford Dictionary of English by an expert lexicographer. This task avoids the issues of a fine granular sense inventory, which provides senses

type	insts	avgSels
target	1108	68.5
noun context	1108	68.5
verb context	591	70.1
adj context	362	37.3
pro context	372	31.9

Table 1: Total word instances for which selectors were acquired (**insts**), and average number of selectors acquired for use in each instance (**avgSels**).

that are difficult even for humans to distinguish. Additionally, considering how recent the event occurred, there is a lot of up-to-date data about the performance of other disambiguation systems to compare with. (Navigli et al., 2007)

Out of 2269 noun, verb, adjective, or adverb instances we are concerned with disambiguating the 1108 noun instances from the 245 sentences in the corpus. These noun instances represent 593 different words. Since we did not use the coarse-grained senses within our algorithm, the predicted senses were correct if they mapped to the correct coarse-grained sense. The average instance had 2.5 possible coarse-grained senses. The average number of selectors acquired for each word is given in Table 1. The bottom of Table 2 shows the random baseline as well as a baseline using the most frequent sense (MFS) heuristic. As previously mentioned, many supervised systems only perform marginally better than the MFS. For the SemEval workshop, only 6 of 15 systems performed better than this baseline on the nouns (Navigli et al., 2007), all of which used MFS as a back off strategy and an external sense tagged data set. Our results are presented as precision (P), recall (R), and F1 value ($F1 = 2 * \frac{P * R}{P + R}$).

4.1 Results and Discussion

Table 2 shows the results when using various similarity for the *target selectors*. We selected gloss-based measures (Banerjee and Pedersen, 2003; Patwardhan et al., 2003) due to the need for handling multiple parts of speech for the *context selectors*. Functionality for our use of many different relatedness measurements was provided by WordNet::Similarity (Pedersen et al., 2004). Our method performs better than the MFS baseline, and clearly better than the random baseline. As one can see, the *scaled_sim* (path2) similarity measure along with the gloss based relatedness

	gloss1	gloss2
path1	78.8	78.3
path2	80.2	78.6
path3	78.7	78.6
IC1	78.6	79.3
IC2	78.5	79.2
IC3	78.0	78.1
gloss1	78.4	80.0
gloss2	78.6	78.9
MFS	baseline	77.4
random	baseline	59.1

Table 2: Performance of our method, given by F1 values (precision = recall), with various similarity measures for *target selectors*: **path1**= sim (normalized depth), **path2** = *scaled_sim*, **path3** = (Wu and Palmer, 1994), **IC1** = (Resnik, 1999), **IC2** = (Lin, 1997), **IC3** = (Jiang and Conrath, 1997), and relatedness measures for *context selectors*: **gloss1** = (Banerjee and Pedersen, 2003), **gloss2** = (Patwardhan et al., 2003). Baselines: **MFS** = most frequent sense, **random** = random choice of sense.

measure of (Banerjee and Pedersen, 2003) gave the best results. Note that the path-based and information content measures, in general, performed equally.

We experimented with using the gloss-based relatedness measures in place of similarity measures. The idea was that one measure could be used for both target selectors and context selectors. As one can gather from the bottom of table 2, for the most part, the measures performed equally. The experimental runtime of the path-based and information content measures was roughly one-fourth that of the gloss-based measures.

Table 3 presents results from experiments where we only attempted to annotate instances with over a minimum number of target selectors (tMin) and context selectors (cMin). We use steps of four for target selectors and steps of ten for context selectors, reflecting a ratio of roughly 2 target selectors for every 5 context selectors. It was more common for an instance to not have any target selectors than to not have context selectors, so we present results with only a tMin or cMin. The main goal of these experiments was simply to determine if the algorithm performed better on instances that we were able to acquire more selectors. We were able to see this was the case as the precision improved at the expense of recall from avoiding the noun instances

tMin	cMin	A	P	R	F1
0	0	1108	80.2	80.2	80.2
4	0	658	84.4	50.1	62.9
16	0	561	85.2	43.1	57.2
0	10	982	81.1	71.9	76.2
0	40	908	81.3	66.6	73.3
4	10	603	85.4	46.4	60.1
8	20	554	85.3	42.6	56.9
12	30	516	86.4	40.2	54.9
16	40	497	86.5	38.8	53.5

Table 3: Number attempted (**A**), Precision (**P**), Recall (**R**) and **F1** values of our method with restrictions on a minimum number of target selectors (**tMin**) and context selectors (**cMin**).

sel	noMFS	1SPD
80.2	79.6	79.8

Table 4: Results of a variety of experiments using *path2* and *gloss1* from the previous table. **noMFS** = no use of most frequent sense, **1SPD** = use of 1 sense per discourse.

that did not have many selectors.

Table 4 shows the results when we modify the method in a few ways. All these results use the *path2* (*scaled_sim*) and *gloss1* (Banerjee and Pedersen, 2003) measures. The results of Table 2 include first sense heuristic used as a back-off strategy for close calls, when multiple senses have a score within 0.05 of each other. Therefore, we experiment without this heuristic presented as *noMFS*, and found our method still performs strongly. We also implemented one sense per discourse, reported as *1SPD*. Our experimental corpus had five documents, and for each document we calculated the most commonly predicted sense and used that for all occurrences of the word within the document. Interestingly, this strategy does not seem to improve the results in our method.

4.2 Comparison with other systems

Table 5 shows the results of our method (sel) compared with a few systems participating in the SemEval coarse-grained all-words task. These results include the median of all participating systems, the top system not using training data (UPV-WSD) (Buscaldi and Rosso, 2007), and the top system using training data (NUS-PT) (Chan et al., 2007). The best performance reported on the

sel	med	UPV-WSD	NUS-PT	SSI
80.2	71.1	79.33	82.31	84.12

Table 5: Comparison of noun F1 values with various participants in the SemEval2007 coarse-grained all-words task.

nouns for the SemEval coarse-grained task, was actually from a system by the authors of the task (SSI) (Navigli and Velardi, 2005). All systems performing better than the MFS used the heuristic as a backoff strategy when unable to output a sense (Navigli et al., 2007). Also, the systems performing better than ours (including SSI) used more sources of sense annotated data.

5 Conclusion

We have presented a method for acquiring knowledge from the Web for noun sense disambiguation. Rather than searching the web with pre-chosen relatives, we search with a string representing the local context of a target word. This produces a list of selectors, words which may replace the target word within its local context. The selectors are then compared with the senses of the target word via similarity and relatedness measures to choose the correct sense. By searching with context instead of simply relatives, we are able to insure more relevant results from the web. Additionally, this method has an advantage over methods which use relatives and context in that it does not restrict the results to include pre-chosen words.

We also show that different types of similarity and relatedness measures are appropriate for different roles in our disambiguation algorithm. We found a path-based measure to be best with *target selectors* while a slower gloss-based method was appropriate for *context selectors* in order to handle multiple POS. For many tasks, information content based measures perform better than path-based measures. However, we found a path-based measure to be just as strong if not stronger in our approach.

Results of our evaluation using the SemEval coarse-grained all-words task showed strength in the use of selectors from the Web for disambiguation. Our system was out-performed only by systems using training data or substantially more annotated data. Future work may improve results through the use of sense tagged corpora, a grammatical parse, or other methods commonly used in

WSD. Additionally, better precision was achieved when requiring a minimum number of selectors, giving promise to improved results with more work in acquiring selectors. This paper has shown an effective and novel method of noun sense disambiguation through the use of selectors acquired from the web.

6 Acknowledgement

This research was supported in part by the NASA Engineering and Safety Center under Grant/Cooperative Agreement NNX08AJ98A.

References

- Agirre, Eneko and David Martinez. 2004. Unsupervised wsd based on automatically retrieved examples: The importance of bias. In *Proceedings of EMNLP 2004*, pages 25–32, Barcelona, Spain, July. Association for Computational Linguistics.
- Banerjee, S. and T. Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 805–810, Acapulco.
- Brody, Samuel, Roberto Navigli, and Mirella Lapata. 2006. Ensemble methods for unsupervised wsd. In *Proceedings of the 21st International Conference on Computational Linguistics*, pages 97–104, Sydney, Australia.
- Budanitsky, Alexander and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.
- Buscaldi, Davide and Paolo Rosso. 2007. Upv-wsd : Combining different wsd methods by means of fuzzy borda voting. In *Proceedings of SemEval-2007*, pages 434–437, Prague, Czech Republic, June.
- Chan, Yee Seng, Hwee Tou Ng, and Zhi Zhong. 2007. Nus-pt: Exploiting parallel texts for word sense disambiguation in the english all-words tasks. In *Proceedings of Proceedings of SemEval-2007*, pages 253–256, Prague, Czech Republic, June.
- Jiang, Jay J. and David W. Conrath. 1997. Semantic similarity on corpus statistics and lexical taxonomy. In *Proceedings of ROCLING X*, Taiwan.
- Leacock, Claudia, Martin Chodorow, and George A. Miller. 1998. Using corpus statistics and wordnet relations for sense identification. *Computational Linguistics*, 24(1):147–165.
- Lin, Dekang. 1997. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of the 35th annual meeting on Association for Computational Linguistics*, pages 64–71.
- Martinez, David, Eneko Agirre, and Xinglong Wang. 2006. Word relatives in context for word sense disambiguation. In *Proceedings of the 2006 Australasian Language Technology Workshop*, pages 42–50.
- McCarthy, Diana, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 279–286, Barcelona, Spain, July.
- Mihalcea, Rada. 2002. Bootstrapping large sense tagged corpora. In *Proceedings of the 3rd International Conference on Languages Resources and Evaluations LREC 2002*, Las Palmas, Spain, May.
- Miller, George, R. Beckwith, Christiane Fellbaum, D. Gross, and K. Miller. 1993. Five papers on wordnet. Technical report, Princeton University.
- Navigli, Roberto and Paola Velardi. 2005. Structural semantic interconnections: A knowledge-based approach to word sense disambiguation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7):1075–1086.
- Navigli, Roberto, Kenneth C. Litkowski, and Orin Hargraves. 2007. Semeval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of SemEval-2007*, pages 30–35, Prague, Czech Republic, June.
- Patwardhan, S., S. Banerjee, and T. Pedersen. 2003. Using Measures of Semantic Relatedness for Word Sense Disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 241–257, Mexico City, Mexico, February.
- Pedersen, T., S. Patwardhan, and J. Michelizzi. 2004. WordNet::Similarity - Measuring the Relatedness of Concepts. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Demonstrations*, pages 38–41, Boston, MA, May.
- Rada, R., H. Mili, E. Bicknell, and M. Blettner. 1989. Development and application of a metric on semantic nets. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 19, pages 17–30.
- Resnik, Philip. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130.
- Wu, Zhibiao and Martha Palmer. 1994. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, New Mexico State University, Las Cruces, New Mexico.
- Yuret, Deniz. 2007. Ku: Word sense disambiguation by substitution. In *Proceedings of SemEval-2007*, pages 207–214, Prague, Czech Republic, June.

Automatic Chinese Catchword Extraction Based on Time Series Analysis

Han Ren¹, Donghong Ji¹, Jing Wan² and Lei Han¹

1 School of Computer Science, Wuhan University 430079, China

2 Center for Study of Language & Information, Wuhan University 430072, China

cslotus@mail.whu.edu.cn, donghong_ji@yahoo.com,

jennifer.wanj@gmail.com, hattason@mail.whu.edu.cn

Abstract

Catchwords refer to those popular words or phrases in a time period. In this paper, we propose a novel approach for automatic extraction of Chinese catchwords. By analyzing features of catchwords, we define three aspects to describe Popular Degree of catchwords. Then we use curve fitting in Time Series Analysis to build Popular Degree Curves of the extracted terms. Finally we give a formula that can calculate Popular Degree values of catchwords and get a ranking list of catchword candidates. Experiments show that the method is effective.

1 Introduction

Generally, a catchword is a term which represents a hot social phenomenon or an important incident, and is paid attention by public society within certain time period. On the one hand, catchwords represent the mass value orientation for a period. On the other hand, they have a high timeliness. Currently, there are quiet a few ranking and evaluations of catchwords every year in various kinds of media. Only in year 2005, tens of Chinese organizations published their ranking list of Chinese catchwords.

Catchwords contain a great deal of information from any particular area, and such words truly and vividly reflect changes of our lives and our society. By monitoring and analysis of catchwords, we can learn the change of public

attention in time. In addition, we may detect the potential changes of some linguistic rules, which can help establish and adjust state language policies.

Currently, two kinds of approaches are adopted to evaluate catchwords. One is by CTR (Click-Through Rate) or retrieval times, but the limitation is that it is just based on frequency, which is only one feature of catchwords. The other is by manual evaluation, but it depends on their subjective judgment to a large extent. In this paper, we propose a novel approach that can automatically analyze and extract Chinese catchwords. By analyzing sample catchwords and finding out their common features, we provide a method to evaluate the popular degree. After ranking, terms that have high values are picked out as catchword candidates.

The rest of the paper is organized as follows. In Section 2, we discuss about the linguistic basis of catchword judgment. In Section 3, we describe the extraction method in detail. In Section 4, we present the experimental results as well as some discussions. Finally, we give the conclusion and future work in Section 5.

2 Linguistic basis

The popularity of a word or phrase contains two factors: time and area, namely how long it lasts and how far it spreads. But neither of them have definite criterion.

2.1 Linguistic definition of catchword

Many researches of catchwords come from pure linguistic areas. Wang (1997) proposed that catchwords, which include words, phrases, sentences or special patterns, are a language form in certain times and among certain groups or communities. Guo (1999) specified that catchwords are popular words, which are widely

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

used in certain period of time among certain groups of people. To sum up, catchwords are a language form spreading quickly within certain area in certain period of time.

According to Zipf's Law (Zipf, 1949), the word that has a higher usage frequency is shorter than others. Catchwords also follow this principle: most catchwords are words and phrases instead of sentences and longer language units, which are more difficult to extract automatically. In the paper, we focus on catchwords as words and phrases.

2.2 Features of catchword

Some features of catchwords have been proposed, but there have been few research to quantify and weigh the features. Zhang (1999) proposed a method to judge catchwords by weighing Circulating Degree of catchwords, which are based on Dynamic Circulating Corpus. But the corpus construction and the judgment still depend on manual efforts.

By analyzing usage frequency of catchwords, we find that being a language phenomenon within a period of time, a catchword has two features: one is high usage frequency, namely a catchword is frequently used in certain period of time; the other is timeliness, namely this situation will lasts for some time. Our quantification method is based on these features.

3 Extraction Method

In this section, the extraction method is described in detail. After term extraction, the features of terms are weighed by time series analysis. The algorithm in section 3.4 shows the process to extract catchword candidates.

3.1 Term Extraction

Catchwords are words or phrases with maximal meanings, most of which are multi-character words or phrases. Word segmentation has a low discrimination for long phrases, while term extraction has a better way to extract them. Zhang (2006) proposed a new ATE algorithm, which is based on the decomposition of prime string. The algorithm evaluates the probability of a long string to be a term by weighing relation degree among sub-strings within the long string. The algorithm can raise the precision in extracting multi-character words and long phrases. In this paper, we use this method to extract terms.

3.2 Popular Degree Curve

For extracted terms, a time granularity should be defined to describe their features. We select 'day' as the time granularity and get every day's usage frequency for each term in one year. These can be described as a time series like below:

$$C_w = \{c_{w1}, c_{w2}, \dots, c_{wt}, \dots, c_{wn}\} \quad (1)$$

C_w is the time series of term w . c_{wt} is the usage frequency of term w in the day t . n is the number of observation days.

As a latent knowledge, two features of catchwords mentioned in section 2.2 exist in their time series. The effective method to find out the latent knowledge in the time series is Time Series Analysis, which includes linear analysis and nonlinear analysis. As the time series of terms belong to nonlinear series, we use nonlinear analysis to deal with them.

After getting usage frequency, we use SMA (Simple Moving Average) method to eliminate the random fluctuation of series C_w . The formula is as follows:

$$\bar{c}_{wt} = \frac{\sum_{j=1}^m c_{w(t-m+j)}}{m} \quad (2)$$

\bar{c}_{wt} is the smoothed usage frequency of term w in the day t and m is the interval. In SMA method, a short interval has a little effect, while a long one may result in low accuracy. So we should specify a proper interval. Through experiments we find that an appropriate interval is between 10 and 20. Smoothed time series is as follows:

$$\bar{C}_w = \{\bar{c}_{w1}, \bar{c}_{w2}, \dots, \bar{c}_{wt}, \dots, \bar{c}_{wn}\} \quad (3)$$

Smoothed time series of terms can be described as curves, in which the coordinate x is day t and coordinate y is \bar{c}_{wt} . Through these curves we can see that, catchwords appear in certain period of time and its usage frequency increases in this period. After reaching the highest point, usage frequency of catchwords decrease slowly. We call this process Popular Degree, which contains three aspects:

1) Popular Trend: the increasing process of usage frequency; the more obviously the popular trend changes, the higher the popular degree is.

2) Peak Value: maximum usage frequency within certain period of time; the larger the peak value is, the higher the popular degree is.

3) Popular Keeping: the decreasing process of usage frequency; the more gently the popular keeping changes, the higher the popular degree is.

Three aspects above determine popular degree of catchwords. Figure 1 shows the smoothed time series curve of the catchword ‘苏丹红²’, evaluated in year 2005:

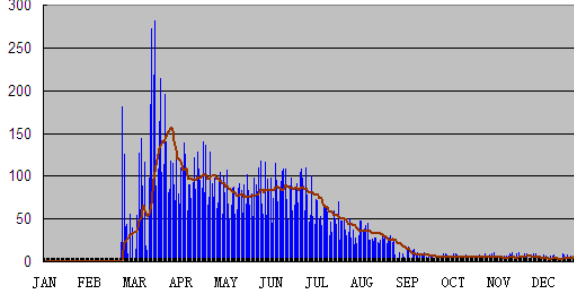


Figure 1. Smoothed time series curve of the catchword ‘苏丹红’

To the catchword ‘苏丹红’, its Popular Trend changes obviously and its Popular Keeping changes gently. Meanwhile, its Peak Value is relatively higher than those of most catchwords. So the catchword ‘苏丹红’ has a high Popular Degree.

According to three aspects of Popular Degree, smoothed time series curve is separated into two parts: one is ascending period, namely Popular Trend process; the other is descending period, namely Popular Keeping process. We use conic fitting to deal with two parts of series. A conic’s formula is like below:

$$Y = a + bt + ct^2$$

According to least square method, a standard equation that can deduce three parameters a, b and c is as follows:

$$\begin{cases} \sum Y = na + b\sum t + c\sum t^2 \\ \sum tY = a\sum t + b\sum t^2 + c\sum t^3 \\ \sum t^2Y = a\sum t^2 + b\sum t^3 + c\sum t^4 \end{cases}$$

Assume T_S is the starting time, T_E is the ending time, and T_M is the time that time series curve reaches the highest point. According to conic fitting method we can get curves of ascending and descending period. Formulas of two conics are as follows:

$$\begin{cases} \varphi(u) = a + bu + cu^2 & T_S \leq t \leq T_M \\ \psi(v) = a' + b'v + c'v^2 & T_M \leq t \leq T_E \end{cases} \quad (4)$$

Variable u and v are usage frequency of a term in a day, $\varphi(u)$ is the formula of ascending curve, and $\psi(v)$ is the formula of descending curve. The curve described by equation (4) is called Popular Degree Curve. Figure 2 shows the Popular Degree Curve of the catchword ‘苏丹红’:

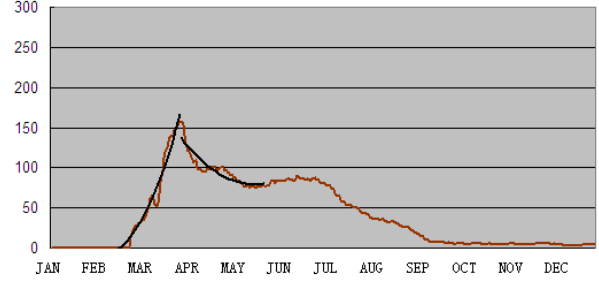


Figure 2. Popular Degree Curve of the catchword ‘苏丹红’

3.3 Popular Degree Value

The decision of catchwords is based on three aspects of Popular Degree described in section 3.2. We propose a formula to calculate Popular Degree values of terms. After getting the values, a ranking list by inverse order is established. The Popular Degree of a catchword is in the direct ratio to its place in the ranking list. The formula is as follows:

$$PD(w) = PT(w) \times PV(w) \times PK(w) \quad (5)$$

$PD(w)$ is the Popular Degree value of the catchword w . $PT(w)$ is the Popular Trend value of w :

$$PT(w) = \alpha \cdot \frac{\varphi(T_M) - \varphi(T_S)}{\varphi(T_M)} \quad (6)$$

α is the adjusting parameter of Popular Trend. The formula indicates that $PT(w)$ is related to changing process of Popular Degree Curve. $PV(w)$ is the Peak Value of w :

$$PV(w) = \beta \cdot \frac{\max\{\bar{c}_{wt}\}}{\frac{1}{N_w} \sum_w \max\{\bar{c}_{wt}\} + \max\{\bar{c}_{wt}\}} \quad (7)$$

β is the adjusting parameter of Peak Value. The formula indicates that $PV(w)$ is related to the maximum usage frequency of w . $PK(w)$ is the Popular Keeping value of w :

$$PK(w) = \gamma \cdot \left(1 - \frac{\psi(T_M) - \psi(T_E)}{\psi(T_M)}\right) \quad (8)$$

γ is the adjusting parameter of Popular Keeping. The formula indicates that $PK(w)$ is related to changing process of Popular Degree Curve. Parameter α , β and γ control proportion of three aspects in Popular Degree value.

² 苏丹红 means Sudan red in English.

All extracted terms are ranked according to their Popular Degree values. Terms that have high scores are picked out as catchword candidates.

3.4 Algorithm

The algorithm of automatic catchwords extraction is described below:

Algorithm Extracting catchwords

Input text collections

Output ranking list of catchword candidates

Method

- 1) use ATE algorithm mentioned in section 3.1 to extract terms
- 2) filter terms that contains numbers and punctuations
- 3) **foreach** term
- 4) calculate its smoothed time series by formula (2)
- 5) use conic fitting method in section 3.2 to get its Popular Degree Curve like equation (4)
- 6) use formula (5) ~ (8) to calculate its Popular Degree value
- 7) rank all Popular Degree values from high to low

4 Experimental Results and Analysis

4.1 Text Collection

In the experiment, we use 136,191 web pages crawled from Sina³'s news reports in year 2005 including six categories: economy, science, current affairs, military, sports and entertainment. For the experimental purpose, we extract body content in every web page by using Noise Reducing algorithm (Shianhua Lin & Janming Ho, 2002). Totally, the extracted subset includes 129,328 documents.

4.2 Experiment settings

In the experiment, several parameters should be settled to perform the catchwords extraction.

• n

A large time granularity may result in low accuracy for conic fitting. In this paper, we select 'day' as the time granularity.

• m

For the interval m in formula (2), a proper value should be specified to not only eliminate random fluctuation but also keep

accuracy of data. In the experiment we find that the proper interval is between 10 and 20.

• T_S and T_E

Catchwords have a high timeliness, so we should specify a time domain. By analysis of sample catchwords, we find that popular time domain for most of them approximately last for not more than 6 months. So we specify the time domain is $n / 2$. Thus the relationship among the starting time T_S and the ending time T_E is below:

$$T_S = T_E - \frac{n}{2}$$

As a proper example, the starting point can be 60 days away from the highest point. Thus the Popular Trend process and the Popular Keeping process both last for nearly 3 months. So the relationship can be described as formulas below:

$$T_S = T_M - \left\lceil \frac{n}{4} \right\rceil, T_E = T_M + \left\lceil \frac{n}{4} \right\rceil$$

• α, β, γ

To keep the Popular Degree values of catchwords within $[0, 1]$, three adjusting parameters are satisfied to the inequation:

$$0 < \alpha, \beta, \gamma \leq 1.$$

Table 1 shows proper values of parameters as schema 1. We also give other schemas, which contain different values of parameters, to compare with the schema 1. In schema 2 to schema 4, default values of parameters are the same with schema 1.

parameter	Value
n	365
t	[1, 365]
m	15
T_S	$T_M - \lceil n / 4 \rceil$
T_E	$T_M + \lceil n / 4 \rceil$
α	1
β	1
γ	1

Table 1. parameters in schema 1

schema 2: different m values

schema 3: different values of T_S and T_E

schema 4: different values of α, β and γ

4.3 Evaluation Measure

Currently, there is no unified standard for catchword evaluation. In year 2005, NLRMRC

³ <http://www.sina.com.cn/>

(National Language Resources Monitoring and Research Centre, held by MOE of China) had published their top 100 Chinese catchwords. We use co-occurrence ratio of catchwords for the evaluation. The formula of co-occurrence ratio is as follows:

$$r = \frac{N_C}{N}$$

N is the number of ranking catchwords. N_C is the co-occurrence of catchwords, namely the number of catchwords which appear both in our approach and NLRMRC in top N .

4.4 Results

We use algorithm described in section 3.4 to get a ranking list of catchword candidates. According to ATE algorithm mentioned in section 3.1, we extract 966,532 terms. After filtering invalid terms we get 892,184 terms and calculate each term's Popular Degree value. Table 2 - 5 shows the co-occurrence ratio with schema 1 - 4.

N=20	N=40	N=60	N=80	N=100
7%	18%	36%	53%	66%

Table 2. Co-occurrence ratio using schema 1

m	N=20	N=40	N=60	N=80	N=100
5	3%	7%	16%	29%	45%
10	4%	11%	25%	44%	59%
20	7%	15%	32%	49%	63%
25	6%	14%	29%	46%	60%

Table 3. Co-occurrence ratio using schema 2

$\frac{T_M - T_S}{T_E - T_M}$	N=20	N=40	N=60	N=80	N=100
1 : 4	0%	3%	8%	15%	22%
2 : 3	4%	14%	30%	49%	64%
3 : 2	5%	15%	33%	51%	63%
4 : 1	2%	5%	12%	21%	26%

Table 4. Co-occurrence ratio using schema 3

	N=20	N=40	N=60	N=80	N=100
$\alpha=0.5$	3%	9%	24%	42%	55%
$\alpha=0.8$	6%	15%	31%	50%	64%
$\beta=0.5$	2%	6%	16%	37%	52%
$\beta=0.8$	5%	13%	29%	47%	59%
$\gamma=0.5$	3%	11%	26%	43%	57%
$\gamma=0.8$	6%	15%	32%	51%	62%

Table 5. Co-occurrence ratio using schema 4

Table 2 shows the co-occurrence ratio of the catchwords extracted by our approach and NLRMRC in top N catchwords ranking list. It indicates that, when N is 100, co-occurrence of the catchwords reaches 66%; when N is lower,

the ratio is also lower. On the one hand, we can see that our approach has a good effect on automatically extracting catchwords, closing to the result of manual evaluation with the increment of N . On the other hand, it proves that divergence exists between our approach and manual evaluation in high-ranking catchwords.

Table 3 indicates that, the condition of $m = 20$ has a better co-occurrence ratio in contrast with others in schema 2. It is because a short interval has a little effect, while a long one may result in low accuracy in SMA.

Table 4 indicates that a better performance can be made when the proportion of $T_M - T_S$ and $T_E - T_M$ is close to 1:1. It proves that Popular Trend process is just as important as Popular Keeping process. Therefore the best time domain of these two processes are both $n / 4$.

Three parameters can adjust the weights of PD , PV and PK in formula (5). Table 5 indicates that three factors above are all important for weighing a catchword, while β is a little more important than α and γ . Therefore, maximum usage frequency of a catchword is a little more important than two other factors.

From Table 2 - 5 we can see that, parameters in schema 1 is most appropriate for the evaluation.

Table 6 shows the ranking list of top 10 catchword candidates according to their Popular Degree values:

candidates ⁴	PD value
苏丹红	0.251262
超级女声	0.220975
油价	0.213843
纺织品谈判	0.196326
TD-SCDMA	0.185691
芙蓉姐姐	0.166730
发现号	0.154803
丁俊晖	0.137211
六方会谈	0.121738
猪链球菌	0.120667

Table 6. Popular Degree values of Top 10 catchword candidates

⁴ 超级女声 means a talent show by Hunan Satellite.

油价 means petroleum price

纺织品谈判 means textile negotiation

芙蓉姐姐 means a famous girl called sister lotus

发现号 means STS Discovery OV-103

丁俊晖 means a billiards player named Junhui Ding

六方会谈 means Six-Party Talks

猪链球菌 means swine streptococcus suis

4.5 Analysis

In our experiment, Popular Values of some catchwords by manual evaluation are lower. By analyzing their time series curves, we find that usage frequencies of these terms are not high. We also find that these catchwords mostly have other expressions. Such as the catchword ‘社会保障体系⁵’ can be also called ‘社保体系⁶’. These two synonyms are treated as one term in manual evaluation that corresponds to promote usage frequency. However, relationship between the two synonyms is not concerned in automatic extraction. They are treated as separate terms. So the Popular Degree Values of these two synonyms are not high either. It proves that parts of catchwords by manual evaluation are collected and generalized. A catchword should be treated not only as a separate word or a phrase, but also as a part of a word-cluster, which consist of synonymous words or phrases. Through word clustering method, we can get an increasing quantity of the co-occurrence of catchwords between our approach and manual evaluations.

5 Conclusions

Being as one aspect of dynamic language research, catchwords have a far-reaching significance for the development of linguistics. The paper proposes an approach that can automatically detect and extract catchwords. By analyzing evaluated catchwords and finding out their common feature called popular degree, the paper provides a method of popular degree quantification and gives a formula to calculate term’s popular degree value. After ranking, terms that have high values are picked out as catchword candidates. The result can be provided as a reference for catchword evaluation. Experiments show that automatic catchword extraction can promote the precision and objectivity, and mostly lighten difficulties and workload of evaluation.

In the experiment, we also find that some catchwords are not isolated, but have a strong relationship and express the same meaning. In the future, we can unite all synonymous catchwords to a word cluster and calculate the cluster’s popular degree value. Thus we would be able to achieve a better performance for extraction.

⁵ 社会保障体系 means social security system

⁶ 社保体系 is the abbreviation of 社会保障体系

Acknowledgement

This work is supported by the Natural Science Foundation of China under Grant Nos.60773011, 60703008.

References

- G.E.P.Box, G.M.Jenkins and G.C.Reinsel. 1994. *Time Series Analysis, Forecasting and Control*. Third Edition, Prentice-Hall.
- Richard L. Burden and J.Douglas Faires. 2001. *Numerical Analysis*. Seventh Edition, Brooks/Cole, Thomson Learning, Inc., pp. 186-226.
- Xi Guo. 1999. *China Society Linguistics*. Nanjing : Nanjing University Press.
- H. Kantz and T. Schreiber. 1997. *Nonlinear Time Series Analysis*. Cambridge University Press, 1997
- Shianhua Lin, Janming Ho. 2002. *Discovering informative content blocks from Web documents*. In: SIGKDD.
- Dechun Wang 1997. *Introduction to Linguistics*. Shanghai: Shanghai Foreign Language Education Press.
- George K.Zipf 1949. *Human Behavior and Principle of Least Effort: an Introduction to Human Ecology*. Addison Wesley, Cambridge, Massachusetts.
- Pu Zhang 1999. *On thinking of language sense and Circulating Degree*. Beijing: Language Teaching and Linguistic Studies, (1).
- Yong Zhang 2006. *Automatic Chinese Term Extraction Based on Decomposition of Prime String*. Beijing: Computer Engineering, (23).

Easy as ABC? Facilitating Pictorial Communication via Semantically Enhanced Layout

Andrew B. Goldberg, Xiaojin Zhu, Charles R. Dyer, Mohamed Eldawy, Lijie Heng

Department of Computer Sciences

University of Wisconsin, Madison, WI 53706, USA

{goldberg, jerryzhu, dyer, eldawy, ljheng}@cs.wisc.edu

Abstract

Pictorial communication systems convert natural language text into pictures to assist people with limited literacy. We define a novel and challenging problem: picture layout optimization. Given an input sentence, we seek the optimal way to lay out word icons such that the resulting picture best conveys the meaning of the input sentence. To this end, we propose a family of intuitive “ABC” layouts, which organize icons in three groups. We formalize layout optimization as a sequence labeling problem, employing conditional random fields as our machine learning method. Enabled by novel applications of semantic role labeling and syntactic parsing, our trained model makes layout predictions that agree well with human annotators. In addition, we conduct a user study to compare our ABC layout versus the standard linear layout. The study shows that our semantically enhanced layout is preferred by non-native speakers, suggesting it has the potential to be useful for people with other forms of limited literacy, too.

1 Introduction

A picture is worth a thousand words—especially when you are someone with communicative disorders, a foreign language speaker, or a young child. Pictorial communication systems aim to automatically convert general natural language text into meaningful pictures. A perfect pictorial

communication system can turn signs and operation instructions into easy-to-understand graphical forms; combined with optical character recognition input, a personal assistant device could create such visual translations on-the-fly without the help of a caretaker. Pictorial communication may also facilitate literacy development and rapid browsing of documents through pictorial summaries.

Pictorial communication research is in its infancy with a spectrum of experimental systems, which we review in Section 2. At one end of the spectrum, some systems render highly realistic 3D scenes but require specific scene-descriptive language. At the other end, some systems perform dictionary-based iconic transliteration (turning words into icons¹ one by one) on arbitrary text but the pictures can be hard to understand. We are interested in using pictorial communication as an assistive communication tool. Thus, our system needs to be able to handle general text yet produce easy-to-understand pictures, which is in the middle of the spectrum. To this end, our system adopts a “collage” approach (Zhu et al., 2007). Given a piece of text (e.g., a sentence), it first identifies important and easy-to-depict words (or phrases) with natural language processing (NLP) techniques. It then finds one good icon per word, either from a manually created picture-dictionary, or via image analysis on image search results. Finally, it lays out the icons to create the picture. Each step involves several interesting research problems.

This paper focuses exclusively on the picture layout component and addresses the following question: Can we use machine learning and NLP techniques to learn a good picture layout that im-

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

¹In this paper, an *icon* refers to a small thumbnail image corresponding to a word or phrase. A *picture* refers to the overall large image corresponding to the whole text.

proves picture comprehension for our target audiences of limited literacy? We first propose a simple yet novel picture layout scheme called “ABC.” Next, we design a Conditional Random Field-based semantic tagger for predicting the ABC layout. Finally, we conduct a user study contrasting our ABC layout to the linear layout used in iconic transliteration. The main contribution of this paper is to introduce the novel task of layout prediction, learned using linguistic features including Prop-Bank role labels, part-of-speech tags, and lexical features.

2 Prior Pictorial Communication Work

At one extreme, there has been significant prior work on “text-to-scene” type systems, which were often intended to aid graphic designers in placing objects in a 3D environment. Example systems include NALIG (Adorni et al., 1983), SPRINT (Yamada et al., 1992), Put (Clay and Wilhelms, 1996), and others (Brown and Chandrasekaran, 1981). Perhaps the best known system of this type, WordsEye (Coyne and Sproat, 2001), uses a large manually tagged collection of 3D polyhedral models to create photo-realistic scenes. Similarly, CarSim (Johansson et al., 2005) can create animated scenes, but operates exclusively in the limited domain of reconstructing road accidents from traffic reports. These systems cater to detailed descriptive text with visual and spatial elements. They are not intended as assistive tools to communicate general text, which is our goal.

Several systems (Zhu et al., 2007; Mihalcea and Leong, 2006; Joshi et al., 2006) attempt to balance language coverage versus picture sophistication. They perform some form of keyword selection, and select corresponding icons automatically from a 2D image database. The result is a pictorial summary representing the main idea of the original text, but precisely determining the original text by looking at the picture can be difficult.

At the other extreme, augmentative and alternative communication software allows users to input arbitrary text. The words, and sometimes common phrases, are semi-automatically transliterated into icons, and displayed in sequential order. Users must learn special icons, which correspond to function words, before the resulting pictures can be fully understood. Examples include SymWriter (Widgit Software, 2007) and Blissymbols (Hehner, 1980).

Other than explicit scene-descriptive languages, pictorial communication systems have not sufficiently addressed the issue of picture layout for general text. We believe a good layout can better communicate the text a picture is trying to convey. The present work studies the use of a semantically inspired layout to enhance pictorial communication. For simplicity, we restrict our attention to the layout of a single sentence. We anticipate the use of text simplification (Chandrasekar et al., 1996; Vickrey and Koller, 2008) to convert complex text into a set of appropriate inputs for our system.

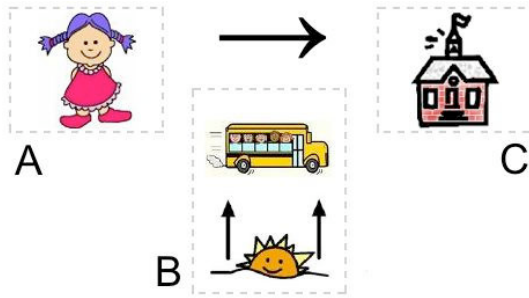
3 The ABC Layout

A good picture layout scheme must be intuitive to humans and easy to generate by computers. To design such a layout, we conducted a pilot study. Five human annotators produced free-hand pictures of many sentences. Analyzing these pictures, we found a large amount of agreement in the use of arrows to mark actions and to provide structure to what would otherwise be a jumble of icons.

Motivated by the pilot study, we propose a simple layout scheme called ABC. It features three *positions*, referred to as A, B, and C. In addition, an arrow points from A through B to C (Figure 1). These positions are meant to denote certain semantic roles: roughly speaking, A denotes “who,” B denotes “what action,” and C denotes “to whom, for what.” Each position can contain any number of icons, each representing a word or phrase in the text. Words that do not play a significant role in the text will be omitted from the ABC layout.

There are two main advantages of the ABC layout:

1. The ABC positioning of icons allows users to infer the semantic role of the corresponding concepts. In particular, we found that verbs can be difficult to depict and understand without such hints. The B position serves as an action indicator to disambiguate between multiple senses of the same icon. For example, in Figure 1, the school bus icon clearly represents the verb phrase “rides the bus,” rather than just the noun “bus.”
2. Such a layout is particularly amenable to machine learning. Specifically, we can turn the problem of finding the optimal layout for an input sentence into a sequence tagging problem, which is well-studied in NLP.



The girl rides the bus to school in the morning
O A B B B O C O O B

Figure 1: Example ABC picture layout, original text, and tag sequence.

3.1 ABC Layout as Sequence Tagging

Given an input sentence, one can assign each word a tag from the set $\{A, B, C, O\}$. The bottom row in Figure 1 shows an example tag sequence. The tag specifies the ABC layout position of the icon corresponding to that word. Tag O means “other” and marks words not included in the picture. Within each position, icons appear in the word order in the input sentence. Therefore, a tag sequence uniquely determines an ABC layout of the picture.

Finding the optimal ABC layout of the input sentence is thus equivalent to computing the most likely tag sequence given the input sentence. We adopt a machine learning approach by training a sequence tagger for this task. To do so, we need to collect labeled training data in the form of sentences with manually annotated tag sequences. We discuss our annotation effort next, and present our machine learning models in Section 4.

3.2 Human Annotated Training Data

We asked the five annotators to manually label 571 sentences compiled from several online sources, including grade school texts about history and science, children’s books, and recent news headlines. Some sentences were written by the annotators and describe daily activities. The annotators tagged each sentence using a Web-based tool to drag-and-drop icons into the desired positions in the layout².

To gauge the quality of the manually labeled data, and to understand the difficulty of the ABC

²The manual tagging actually employs a more detailed tag set to denote phrase structure: Each A, B, or C tag is combined with a modifier of *b* (begin phrase) or *i* (inside phrase). For example, the phrase “rides the bus” in Figure 1 is tagged with $B_b B_i B_i$, and shares one icon. The icons were also manually selected by the annotator from a list of Web image search results.

layout, we computed inter annotator agreement among three of the five annotators on a common set of 48 sentences. Considering all pair-wise comparisons of the three annotators, the overall average tag agreement was 77%. This measures the total number of matching tags (across all sentences) divided by the total number of tags. Matching strictly requires both the correct tag and the correct modifier. We also computed Fleiss’ kappa, which measures the degree of inter-annotator agreement beyond the amount expected by chance (Fleiss, 1971). The values range from 0 to 1, with 1 indicating perfect agreement. The kappa statistic was 0.71, which is often considered moderate to high agreement.

Further inspection revealed that most disagreement was due to annotators reversing A and C tags. This could arise from interpreting passive sentences in different ways or trying to represent physical movement. For example, some annotators found it more natural to depict eating by placing a food item in A and the eater in C, treating the arrow as the transfer of food. It was also common for annotators to disagree on whether certain adverbs and time modifiers belong in B or in C. These differences all suggest the highly subjective nature of conceptualizing pictures from text.

4 A Conditional Random Field Model for ABC Layout Prediction

We now introduce our approach to automatically predicting the ABC layout of an input sentence. While it was most natural for human annotators to annotate text at the word level, early experiments quickly revealed that predicting tags at this level is quite challenging. Most of this stems from the fact that human annotators tend to fragment the text into many small segments based on the availability of good icons. For example, the phrase “the white pygmy elephant” may be tagged as “O A O A” because it is difficult for the annotator to find an icon of this exact phrase or the word “pygmy,” but easy to find icons of “white” and “elephant” separately. Essentially, human annotation combines two tasks in one: deciding where each phrase goes in the layout, and deciding which words within a phrase can be depicted with icons.

To rectify this situation, we make layout predictions at the level of chunks (phrases); that is, we automatically break the text into chunks, then predict one A, B, C, or O tag for each chunk. Since the

tag choices made for different chunks may depend on each other, we employ Conditional Random Fields (CRF) (Lafferty et al., 2001), which are frequently used in sequential labeling tasks like information extraction. Our choice of chunking is described in Section 4.1, and the CRF models and input features are described in Section 4.2. The task of deciding which words within a chunk should appear in the picture is addressed by a “word picturability” model, and is discussed in a separate paper.

For training, we automatically map the word-level tags in our annotated data to chunk-level tags based on the majority ABC tag within a chunk.

4.1 Chunking by Semantic Role Labeling

Ideally, we would like semantically coherent text chunks to be represented pictorially in the same layout position. To obtain such chunks, we leverage existing semantic role labeling (SRL) technology (Palmer et al., 2005; Gildea and Jurafsky, 2002). SRL is an active NLP task in which words or phrases in a sentence are assigned a label indicating the role they play with respect to a particular verb (also known as the target predicate). SRL systems like FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005) aim to provide a rich representation for applications requiring some degree of natural language understanding, and are thus perfectly suited for our needs. We shall focus on PropBank labels because they are easier to use for our task. To obtain semantic role labels, we use the automatic statistical semantic role labeler ASSERT (Pradhan et al., 2004), trained to identify PropBank arguments through the use of support vector machines and full syntactic parses.

To understand how SRL can be useful for deriving pictorial layouts, consider the sentence “The boy gave the ball to the girl.” PropBank marks the semantic role labels of the “arguments” of verbs. The target verb “give” is part of the frameset “transfer,” with core arguments “Arg0: giver” (the boy), “Arg1: thing given” (the ball), and “Arg2: entity given to” (the girl). Verbs can also involve non-core modifier arguments, such as ArgM-TMP (time), ArgM-LOC (location), ArgM-CAU (cause), etc. The entities playing semantic roles are likely to be entities we want to portray in a picture. For PropBank, Arg0 often represents an Agent, and Arg1 the Patient or Theme. If we could map the different semantic role labels to ABC tags with simple rules, then we would be done.

Unfortunately, it is not this simple, as PropBank roles are verb-specific. As Palmer et al. pointed out, “No consistent generalizations can be made across verbs for the higher-numbered arguments” (Palmer et al., 2005). In the above example, we might expect a layout rule of [Arg0]→A, [Target, Arg1]→B, [Arg2]→C. However, this rule does not generalize to other verbs, such as “drive,” as in the sentence “The boy drives his parents crazy,” which also has three core arguments “Arg0: driver,” “Arg1: thing in motion,” and “Arg2: secondary predication on Arg1.” However, here the action is figurative, and we would expect a layout rule that puts Arg1 in position C: [Arg0]→A, [Target]→B, [Arg1,Arg2]→C.

In addition, while modifier arguments have the same meaning across verbs, their pictorial representation may differ based on context. Consider the sentences “Polar bears live in the Arctic.” and “Yesterday at the zoo, the students saw a polar bear.” In the former, a human annotator is likely to place an icon for the ArgM-LOC “in the Arctic” in position C (e.g., following a polar bear icon in A and a house icon in B). However, the ArgM-LOC in the second sentence, “at the zoo,” seems more appropriately placed in position B since it describes where this particular action occurred.

Finally, the situation is further complicated when a sentence contains multiple verbs. SRL treats each verb in isolation, producing multiple sets of role labels, yet our goal is to produce a single picture. Clearly, the mapping from semantic roles to layout positions is non-trivial. We describe our statistical machine learning approach next.

4.2 Our CRF Models and Features

We use a linear-chain CRF as our sequence tagging model. A CRF is a discriminative model of the conditional probability $p(\mathbf{y}|\mathbf{x})$, where \mathbf{y} is the sequence of layout tags in $\mathcal{Y} = \{A, B, C, O\}$, and \mathbf{x} is the sequence of SRL chunks produced by the process described in Section 4.1. Our CRF has the general form

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \left(\sum_{t=1}^{|\mathbf{x}|} \sum_{k=1}^K \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}, t) \right)$$

where the model parameters are $\{\lambda_k\}$. We use binary features $f_k(y_t, y_{t-1}, \mathbf{x}, t)$ detailed below. Finally, we use an isotropic Gaussian prior $N(0, \sigma^2 I)$ on parameters as regularization.

We explored three versions of the above model by specializing the weighted feature function $\lambda_k f_k(\cdot)$. **Model 1** ignores the pairwise label potentials and treats each labeling prediction independently: $\lambda_{jk} \mathbf{1}_{\{y_t=j\}} f_k(\mathbf{x}, t)$, where $\mathbf{1}_{\{z\}}$ is an indicator function on z . This is equivalent to a multi-class logistic regression classifier. **Model 2** resembles a Hidden Markov Model (HMM) by factoring pairwise label potentials and emission potentials: $\lambda_{ij} \mathbf{1}_{\{y_{t-1}=i\}} \mathbf{1}_{\{y_t=j\}} + \lambda_{jk} \mathbf{1}_{\{y_t=j\}} f_k(\mathbf{x}, t)$. Finally, **Model 3** has the most general linear-chain potential: $\lambda_{ijk} \mathbf{1}_{\{y_{t-1}=i\}} \mathbf{1}_{\{y_t=j\}} f_k(\mathbf{x}, t)$. Model 3 is the most flexible, but has the most weights to learn.

We use the following binary predicate features $f_k(\mathbf{x}, t)$ in all our models, evaluated on each chunk produced by the semantic role labeler:

1. PropBank role label(s) of the chunk (e.g., Target, Arg0, Arg1, ArgM-LOC). A chunk can have multiple role labels if the sentence contains multiple verbs; in this case, we merge the multiple SRL results by taking their union.

2. Part-of-speech tags of all the words in the chunk. All syntactic parsing results are obtained from the Stanford Parser (Klein and Manning, 2003), using the default PCFG model.

3. Phrase type (e.g., NP, VP, PP) of the deepest syntactic parse tree node covering the entire chunk. We also include a feature indicating whether the phrase is nested within an ancestor VP.

4. Lexical features: individual word identities in the top 5000 most frequent words in the Google 1T 5gram corpus (Brants and Franz, 2006). For other words, we use their automatically predicted WordNet supersenses (Ciaramita and Altun, 2006). Supersenses are 41 broad semantic categories (e.g., noun.location, verb.communication). By dividing lexical features in this way, we hope to learn specific qualities of common words, but generalize across rarer words.

We also experimented with features derived from typed dependency relations, but these did not improve our models. We suspect the PropBank role labels capture much of the same information. In addition, the Google 5000-word list was the best among several word lists that we explored for splitting up the lexical features.

4.3 CRF Experimental Results

We trained our CRF models using the MALLET toolkit (McCallum, 2002). Our complete dataset consists of the 571 manually annotated sen-

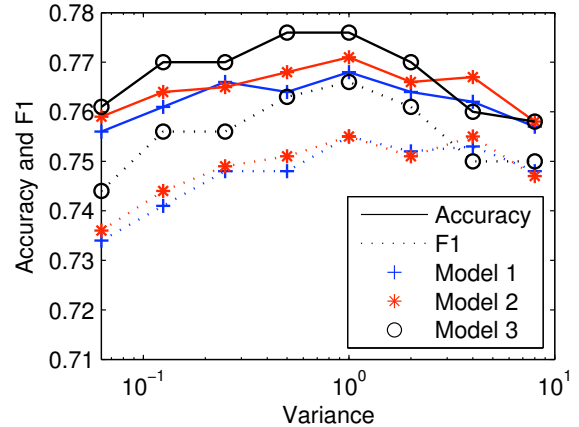


Figure 2: 5-fold cross validation results for different values of the regularization parameter (variance σ^2) and three CRF models predicting A, B, C, or O layout tags.

tences (tags mapped to chunk-level). The only tuning parameter is the Gaussian prior variance, σ^2 . We performed 5-fold cross validation, varying σ^2 and comparing performance across models. Figure 2 demonstrates that peak per-chunk accuracy (77.6%) and macro-averaged F1 scores are achieved using the most general sequence labeling model. As a result, the user study in the next section is based on layouts predicted by Model 3 with $\sigma^2 = 1.0$, trained on all the data.

To understand which features contribute most to performance, we experimented with removing each of the four types (individually). Peak accuracy drops the most when lexical features are removed (76.4%), followed by PropBank features (76.5%), phrase features (76.9%), and POS features (77.1%).

The features in the final learned model make intuitive sense. It prefers tag transitions $A \rightarrow B$ and $B \rightarrow C$, but not $A \rightarrow C$ or $C \rightarrow A$. The model likes the word “I” and noun phrases (not nested in a verb phrase) to have tag A. Verbs and ArgM-NEGs are frequently tagged B, while noun.object’s, Arg4s, and ArgM-CAUs are typically C. The model discourages Arg0s and conjunctions in B, and dislikes adverbial phrases and noun.time’s in C.

While 77.6% cross validation accuracy may seem low, it is in fact close to the 81% inter-annotator agreement³, and thus close to optimal. The confusion matrix (not shown) reveals that most er-

³The 81% agreement is on mapped chunk-level tags without modifiers (Fleiss’ kappa 0.74), while the 77% agreement in Section 3.2 is on word-level tags with modifiers.

rors probably arise from disagreements in the individual annotators. The most common errors are predicting B for chunks labeled O and confusing tags B and C. Manually inspecting the pictures in our training set shows that annotators often omitted the verb (such as “is” or “has”) and left the B position empty, since it could be inferred by the presence of the arrow and the images in A and C. Also, annotators tended to disagree on the location of adverbial expressions, dividing them between positions B and C. Finally, only 3.3% of chunks were incorrectly omitted from the pictures. Therefore, we conclude that our CRF models are capable of predicting the ABC layouts.

5 User Study

We have proposed the ABC layout, and showed that we can learn to predict it reasonably well. But an important question remains: *Can the proposed ABC layout help a target audience of limited literacy understand pictures better, compared to the linear layout used in state-of-the-art augmentative and alternative communication software?* We describe a user study as our first attempt to answer this question. This line of work has two main challenges: one is the practical difficulty of working with human subjects of limited literacy; the other is the lack of a quantitative measure of picture comprehension.

[Subjects]: To partially overcome the first challenge, we recruited two groups of subjects with medium and high literacy respectively, in hopes of extrapolating our findings towards the low literacy group. Specifically, the medium group consisted of seven non-native English speakers who speak some degree of English—“medium literacy” refers to their English fluency; twelve native English speakers comprised the high literacy group. All subjects were adults and did not include the authors of this paper or the five annotators. The subjects had no prior exposure to pictorial communication systems.

[Material]: We randomly chose 90 test sentences from three sources⁴ representing our target application domains: short narratives written by and for individuals with communicative disorders (symbolworld.org); one-sentence news synopses written in simple English targeting foreign language learners (simpleenglishnews.com); and the child

writing sections of the LUCY corpus (Sampson, 2003). We created two pictures for each test sentence: one using a linear layout and one using an ABC layout. For the linear layout, we used SymWriter. Typing text in SymWriter automatically produces a left-to-right sequence of icons, chosen from an icon database. In cases where SymWriter suggests several possible icons for a word, we manually selected the best one. For words not in the database, we found appropriate thumbnail images using Web image search. This is how a typical user would use SymWriter. To produce the ABC layout, we applied the trained CRF tagger Model 3 to the test sentence. After obtaining A, B, C, and O tags for text chunks, we placed the corresponding icons (from SymWriter’s linear layout) in the correct layout positions. Icons for words tagged O did not appear in the ABC version of the picture. Aside from this difference, both pictures of each test sentence contained exactly the same icons—the only difference was the layout.

[Protocol]: All 19 subjects observed each of the 90 test sentences exactly once: 45 with the linear layout and 45 with the ABC layout. The layouts and the order of sentences were both randomized throughout the sequence, and the subjects were counter-balanced so each sentence’s linear and ABC layouts were viewed by roughly equal numbers of subjects. At the start of the study, each subject read a brief introduction describing the task and saw an example of each layout style. Then for each test sentence, we displayed a picture, and the subject typed a guess of the underlying sentence. Finally, the subject provided a confidence rating (2=“almost sure,” 1=“maybe correct,” or 0=“no idea”). We measured response time as the time from image display until sentence/rating submission. Figure 3 shows a test sentence in both layouts, together with several subjects’ guesses.

[Evaluation metrics]: As noted above, the second main challenge is measuring picture comprehension—we need a way to compare the original sentences with the subjects’ guesses. In many ways, this is like machine translation (via pictures), so we turned to two automatic evaluation metrics: BLEU-1 (Papineni et al., 2002) and METEOR (Lavie and Agarwal, 2007). BLEU-1 computes unigram precision (i.e., fraction of response words that exactly match words in the original), multiplied by a brevity penalty for omit-

⁴Distinct from the sources of the 571 training sentences.

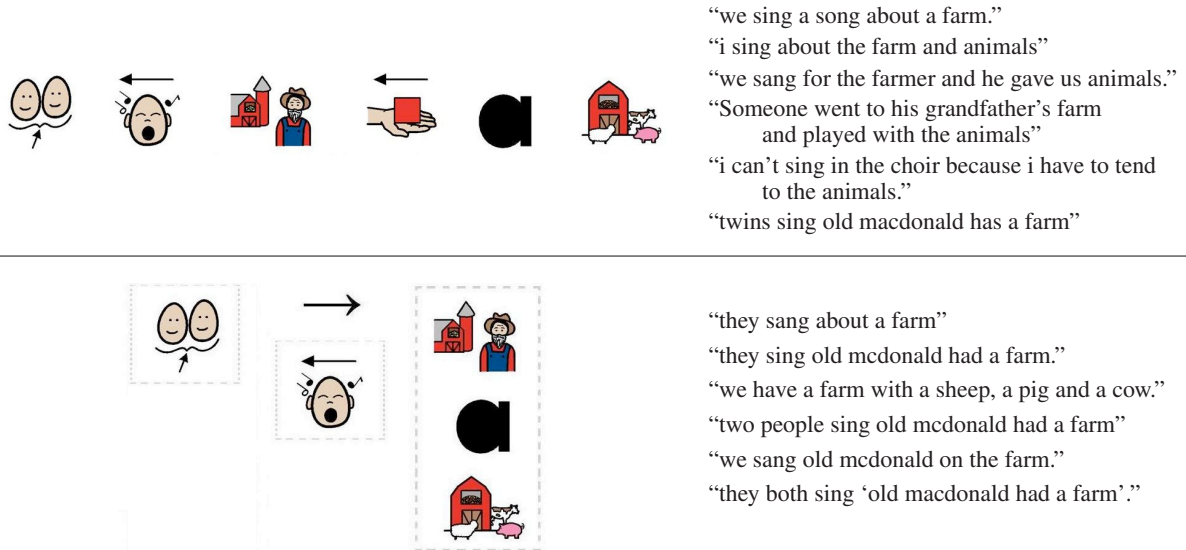


Figure 3: The linear and ABC layout pictures for the test sentence “We sang Old MacDonald had a farm.” and some subjects’ guesses. Note the predicted ABC layout omits the ambiguous “had” icon.

ting words. In contrast, METEOR finds a one-to-one word alignment between the texts that allows partial matches (after stemming and by considering WordNet-based synonyms) and optionally ignores stop words. Based on this alignment, unigram precision, recall, and weighted F measure are computed, and the final METEOR score is obtained by scaling F to account for word-order preservation. We computed METEOR using its default parameters and the stop word list from the Snowball project (Porter, 2001).

[Results]: We report average METEOR and BLEU scores, confidence ratings, and response time for the 4 conditions (native vs. non-native, ABC vs. linear) in Table 1. The most striking observation is that native speakers perform better (in terms of METEOR and BLEU) with the linear layout, while non-native speakers do better with ABC.⁵

To explain this finding, it is worth noting that SymWriter pictures include function words, whose icons are abstract but distinct. We speculate that even though none of our subjects were trained to recognize these function-word icons, the native speakers are more accustomed to the English syntactic structure, so they may be able to transliterate those icons back to words. In an ABC lay-

⁵Using a Mann-Whitney rank sum test, the difference in native speakers’ METEOR scores is statistically significant ($p = 0.003$), though the other differences are not (native BLEU, $p = 0.085$; non-native METEOR, $p = 0.172$; non-native BLEU, $p = 0.170$). Nevertheless, we observe some evidence to support our hypothesis that non-native speakers benefit from the ABC layout, and we intend to conduct follow-up experiments to test the claim further.

	Non-native		Native	
	ABC	Linear	ABC	Linear
METEOR	0.1975	0.1800	0.2955	0.3335
BLEU	0.1497	0.1456	0.2710	0.3011
Conf.	0.50	0.47	0.90	0.89
Time	47.4s	47.8s	38.1s	38.6s

Table 1: User study results.

out, the sentence order is mostly removed, and some phrases might be omitted due to the O tag. Thus native speakers do not get as many syntactic hints. On the other hand, non-native speakers do not have the same degree of built-in English syntactic knowledge. As such, they do not gain much from seeing the whole sentence sequence including function-word icons. Instead, they may have benefited from the ABC layout’s added organization and potential exclusion of irrelevant icons.

If this reasoning holds, it has interesting implications for viewers who have lower English literacy: they might take away more meaning from a semantically structured layout like ABC. Verifying this is a direction for future work.

Finally, it is interesting that all subjects feel more confident in their responses to ABC layouts than linear layouts, and, despite their added complexity, ABC layouts do not require more response time than linear layouts.

6 Conclusions

We proposed a semantically enhanced picture layout for pictorial communication. We formulated our ABC layout prediction problem as sequence tagging, and trained CRF models with linguistic features including semantic role labels. A user study indicated that our ABC layout has the potential to facilitate picture comprehension for people with limited literacy. Future work includes incorporating ABC layouts into our pictorial communication system, improving other components, and verifying our findings with additional user studies.

Acknowledgments

This work is supported by NSF IIS-0711887, and by the Wisconsin Alumni Research Foundation.

References

- Adorni, G., M. Di Manzo, and G. Ferrari. 1983. Natural language input for scene generation. In *ACL*.
- Baker, C. F., C. J. Fillmore, and J. B. Lowe. 1998. The Berkeley FrameNet Project. In *COLING*.
- Brants, T. and A. Franz. 2006. Web 1T 5-gram version 1.1. Linguistic Data Consortium, Philadelphia.
- Brown, D. C. and B. Chandrasekaran. 1981. Design considerations for picture production in a natural language graphics system. *SIGGRAPH*, 15(2).
- Chandrasekar, R., C. Doran, and B. Srinivas. 1996. Motivations and methods for text simplification. In *COLING*.
- Ciaramita, M. and Y. Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *EMNLP*.
- Clay, S. R. and J. Wilhelms. 1996. Put: Language-based interactive manipulation of objects. *IEEE Computer Graphics and Applications*, 16(2).
- Coyne, B. and R. Sproat. 2001. WordsEye: An automatic text-to-scene conversion system. In *SIGGRAPH*.
- Fleiss, J. L. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5).
- Gildea, D. and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3).
- Hehner, B. 1980. *Blissymbols for use*. Blissymbolics Communication Institute.
- Johansson, R., A. Berglund, M. Danielsson, and P. Nuges. 2005. Automatic text-to-scene conversion in the traffic accident domain. In *IJCAI*.
- Joshi, D., J. Z. Wang, and J. Li. 2006. The story picturing engine—a system for automatic text illustration. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2(1).
- Klein, D. and C. D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*.
- Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.
- Lavie, A. and A. Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Second Workshop on Statistical Machine Translation*, June.
- McCallum, A. K. 2002. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>.
- Mihalcea, R. and B. Leong. 2006. Toward communicating simple sentences using pictorial representations. In *Association of Machine Translation in the Americas*.
- Palmer, M., D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).
- Papineni, K., S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.
- Porter, M. F. 2001. Snowball: A language for stemming algorithms. <http://snowball.tartarus.org/>.
- Pradhan, S., W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *HLT/NAACL*.
- Sampson, G. 2003. The structure of children's writing: Moving from spoken to adult written norms. In Granger, S. and S. Petch-Tyson, editors, *Extending the Scope of Corpus-Based Research*. Rodopi.
- Vickrey, D. and D. Koller. 2008. Sentence simplification for semantic role labeling. In *ACL*. To appear.
- Widgit Software. 2007. SymWriter. <http://www.mayer-johnson.com>.
- Yamada, A., T. Yamamoto, H. Ikeda, T. Nishida, and S. Doshita. 1992. Reconstructing spatial image from natural language texts. In *COLING*.
- Zhu, X., A. B. Goldberg, M. Eldawy, C. Dyer, and B. Strook. 2007. A Text-to-Picture synthesis system for augmenting communication. In *AAAI*.

A Nearest-Neighbor Approach to the Automatic Analysis of Ancient Greek Morphology

John Lee

Spoken Language Systems
MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA 02139, USA
jsylee@csail.mit.edu

Abstract

We propose a data-driven method for automatically analyzing the morphology of ancient Greek. This method improves on existing ancient Greek analyzers in two ways. First, through the use of a nearest-neighbor machine learning framework, the analyzer requires no hand-crafted rules. Second, it is able to predict novel roots, and to rerank its predictions by exploiting a large, unlabelled corpus of ancient Greek.

1 Introduction

The civilization of ancient Greece, from which the Western world has received much of its heritage, has justly received a significant amount of scholarly attention. To gain a deeper understanding of the civilization, access to the essays, poems, and other Greek documents in the original language is indispensable.

Ancient Greek is a highly inflected Indo-European language¹. A verb, for example, is inflected according to its person, number, voice, tense/aspect and mood. According to (Crane, 1991), “a single verb could have roughly 1,000 forms, and, if we consider that any verb may be preceded by up to three distinct prefixes, the number of forms explodes to roughly 5,000,000.” The inflections are realized by prefixes and suffixes to

the stem, and sometimes spelling changes within the stem. These numerous forms can be further complicated by accents, and by additional spelling changes at morpheme boundaries for phonological reasons. The overall effect can yield an inflected form in which the root² is barely recognizable.

Indeed, a staple exercise for students of ancient Greek is to identify the root form of an inflected verb. This skill is essential; without knowing the root form, one cannot understand the meaning of the word, or even look it up in a dictionary.

For Classics scholars, these myriad forms also pose formidable challenges. In order to search for occurrences of a word in a corpus, all of its forms must be enumerated, since words do not frequently appear in their root forms. This procedure becomes extremely labor-intensive for small words that overlap with other common words (Crane, 1991).

Automatic morphological analysis of ancient Greek would be useful for both educational and research purposes. In fact, one of the first analyzers was developed as a pedagogical tool (Packard, 1973). Today, a widely used analyzer is embedded in the Perseus Digital Library (Crane, 1996), an internet resource utilized by both students and researchers.

This paper presents an analyzer of ancient Greek that infers the root form of a word. It introduces two innovations. First, *it utilizes a nearest-neighbor framework* that requires no hand-crafted rules, and provides analogies to facilitate learning.

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

¹All Greek words are transcribed into the Roman alphabet in this paper. The acute, grave and circumflex accents are represented by diacritics, as in *ó*, *ò* and *õ*, respectively. Smooth breathing marks are omitted; rough breathing marks are signalled by *h*. Underbars used in *ε* and *ο* represent eta and omega.

²The root is also called the “base” or “lexical look-up” form, since it is the form conventionally used in dictionary entries. For verbs in ancient Greek, the root form is the first person singular present active indicative form. (cf. for English, it is the infinitive.) For nouns, it is the nominative singular form. For adjectives, it is the nominative singular masculine form.

Person/Num	Form	Person/Num	Form
1st/singular	<i>lúo</i>	1st/plural	<i>lúomen</i>
2nd/singular	<i>lúeis</i>	2nd/plural	<i>lúete</i>
3rd/singular	<i>lúei</i>	3rd/plural	<i>lúousi(n)</i>

Table 1: Paradigm table for the present active indicative verb. It uses as example the verb *lúo* (“to loosen”), showing its inflections according to person and number.

Second, and perhaps more significantly, *it exploits a large, unlabelled corpus to improve the prediction of novel roots.*

The rest of the paper is organized as follows. We first motivate these innovations (§2) and summarize previous research in morphological analysis (§3). We then describe the data (§4) and our adaptations to the nearest-neighbor framework (§5-6), followed by evaluation results (§7).

2 Innovations

2.1 Use of Analogy and Nearest Neighbor

Typically, a student of ancient Greek is expected to memorize a series of “paradigms”, such as the one shown in Table 1, which can fill several pages in a grammar book. Although the paradigm table shows the inflection of only one particular verb, *lúo* (“to loosen”), the student needs to apply the patterns to other verbs. In practice, rather than abstracting the patterns, many students simply memorize these “paradigmatic” verbs, to be used as analogies for identifying the root form of an unseen verb. Suppose the unseen verb is *phéreis* (“you carry”); the reasoning would then be, “I know that *lúeis* is the second person singular form of the root *lúo*; similarly, *phéreis* must be the second person singular form of *phéro*.”

The use of analogy can be especially useful when dealing with a large number of rules, for example with the so-called “contract verbs”. The stem of a contract verb ends in a vowel; when a vowel-initial suffix is attached to the stem, spelling changes occur. For instance, the stem *plero-* (“to fill”) combined with the suffix *-omen* becomes *pler-oũ-men*, due to interaction between two omicrons at the boundary. While it is possible to derive these changes from first principles, or memorize the rules for all vowel permutations (e.g., “o” + “o” = “oũ”), it might be easier to recall the spelling changes seen in a familiar verb (e.g., *pleroo* → *pleroũmen*), and then use analogy to infer the root

of an unseen verb.

The nearest-neighbor machine learning framework is utilized to provide these analogies. Given a word in an inflected form (e.g., *phéreis*), the algorithm searches for the root form (*phéro*) among its “neighbors”, by making substitutions to its prefix and suffix. Valid substitutions are to be harvested from pairs of inflected and root forms (e.g., $\langle lúeis, lúo \rangle$) in the training set; these pairs, then, can serve as analogies to reinforce learning.

Furthermore, these affix substitutions can be learned automatically, reducing the amount of engineering efforts. They also increase the transparency of the analyzer, showing explicitly how it derives the root.

2.2 Novel Roots

Ancient Greek, in its many dialects, has been used from the time of Homer to the Middle Ages, in texts of a wide range of genres. Even the most comprehensive dictionaries do not completely cover its extensive vocabulary. To the best of our knowledge, all existing analyzers for ancient Greek require a pre-defined database of stems; thus, they are likely to run into words with unknown or novel roots, which they are not designed to analyze.

Rather than expanding an existing database to increase coverage, we create a mechanism to handle all novel roots. Since words do not often appear in their root forms, inferring a novel root from a surface form is no easy task (Lindén, 2008). We propose the use of unlabelled data to guide the determination of a novel root.

3 Previous Work

After a brief discussion on morphological analysis in general, we will review existing analyzers for ancient Greek in particular.

3.1 Morphological Analysis

A fundamental task in morphological analysis is the segmentation of a word into morphemes, that is, the smallest meaningful units in the word. Unsupervised methods have been shown to perform well in this task. In the recent PASCAL challenge, the best results were achieved by (Keshava and Pitler, 2006). Their algorithm discovers affixes by considering words that appear as substrings of other words, and by estimating probabilities for morpheme boundaries. Another successful ap-

proach is the use of Minimum Description Length, which iteratively shortens the length of the morphological grammar (Goldsmith, 2001).

Spelling changes at morpheme boundaries (e.g., *deny* but *deni-al*) can be captured by orthographic rules such as “change *y-* to *i-* when the suffix is *-al*”. Such rules are specified manually in the two-level model of morphology (Koskenniemi, 1983), but they can also be induced (Dasgupta, 2007). Allomorphs (e.g., “*deni*” and “*deny*”) are also automatically identified in (Dasgupta, 2007), but the general problem of recognizing highly irregular forms is examined more extensively in (Yarowsky and Wicentowski, 2000). They attempt to align every verb to its root form, by exploiting a combination of frequency similarity, context similarity, edit distance and morphological transformation probabilities, all estimated from an unannotated corpus. An accuracy of 80.4% was achieved for highly irregular words in the test set.

3.2 Challenges for Ancient Greek

Ancient Greek presents a few difficulties that prevent a naive application of the minimally supervised approach in (Yarowsky and Wicentowski, 2000). First, frequency and context analyses are sensitive to data sparseness, which is more pronounced in heavily inflected languages, such as Greek, than in English. Many inflected forms do not appear more than a few times. Second, many root forms do not appear³ in the corpus. In Finnish and Swahili, also highly inflected languages, only 40 to 50% of words appear in root forms (Lindén, 2008). The same may be expected of ancient Greek.

Indeed, for these languages, predicting novel roots is a challenging problem. This task has been tackled in (Adler et al., 2008) for modern Hebrew, and in (Lindén, 2008) for Finnish. In the former, features such as letter *n*-grams and word-formation patterns are used to predict the morphology of Hebrew words unknown to an existing analyzer. In the latter, a probabilistic approach is used for harvesting prefixes and suffixes in Finnish words, favoring the longer ones. However, no strategy was proposed for irregular spelling in stems.

³The root forms of contract verbs, e.g. *pleróō*, are not even inflected forms.

Surface Form	Morphological Annotation	Root Form
<i>kaì</i> (<i>and</i>)	Conjunction	<i>kaí</i>
<i>pneûma</i> (<i>spirit</i>)	Noun 3rd decl	<i>pneûma</i>
<i>theoû</i> (<i>God</i>)	Noun 2nd decl	<i>theós</i>
<i>epephéreto</i> (<i>hover</i>)	Verb	<i>phéro</i>

Table 2: Sample data from parts of Genesis 1:2 (“and the Spirit of God was hovering over ...”). The original annotation is more extensive, and only the portion utilized in this research is shown here.

3.3 Ancient Greek Morphological Analysis

The two most well-known analyzers for ancient Greek are both rule-based systems, requiring *a priori* knowledge of the possible stems and affixes, which are manually compiled. To give a rough idea, some 40,000 stems and 13,000 inflections are known by the MORPHEUS system, which will be described below.

The algorithm in MORPH (Packard, 1973) searches for possible endings that would result in a stem in its database. If unsuccessful, it then attempts to remove prepositions and prefixes from the beginning of the word. Accents, essential for disambiguation in some cases, are ignored. The analyzer was applied on Plato’s *Apology* to study the distribution of word endings, for the purpose of optimizing the order of grammar topics to be covered in an introductory course. Evaluation of the analyzer stressed this pedagogical perspective, and the accuracy of the analyses is not reported.

MORPHEUS (Crane, 1991) augments MORPH with a generation component which, given a stem, enumerates all possible inflections in different dialects, including accents. When accents are considered during analysis, the precision of the analyzer improves by a quarter. However, the actual precision and the test set are not specified.

In this paper, we have opted for a data-driven approach, to automatically determine the stems and affixes from training data.

4 Data

4.1 Morphology Data

We used the Septuagint corpus⁴ prepared by the Center for Computer Analysis of Texts at the University of Pennsylvania. The Septuagint, dating from the third to first centuries BCE, is a

⁴<http://ccat.sas.upenn.edu/gopher/text/religion/biblical/>

Part-of-speech	Percent
Verbs	68.6%
Adjectives	10.4%
Nouns (1st declension)	5.6%
Nouns (2nd declension masculine)	4.3%
Nouns (2nd declension neuter)	2.8%
Nouns (3rd declension)	7.6%
other	0.7%

Table 3: Statistics on the parts-of-speech of the words in the test set, considering only unique words.

Greek translation of the Hebrew Bible. The corpus is morphologically analyzed, and Table 2 shows some sample data.

The corpus is split into training and test sets. The training set is made up of the whole Septuagint except the first five books. It consists of about 470K words, with 37,842 unique words. The first five books, also known as the Torah or Pentateuch, constitute the test set. It contains about 120K words, of which there are 3,437 unique words not seen in the training set, and 7,381 unique words seen in training set. A breakdown of the parts-of-speech of the test set is provided in Table 3. Proper nouns, many of which do not decline, are excluded from our evaluation.

4.2 Unlabelled Data

To guide the prediction of novel roots, we utilize the *Thesaurus Linguae Graecae* (Berkowitz and Squitter, 1986) corpus. The corpus contains more than one million unique words, drawn from a wide variety of ancient Greek texts.

4.3 Evaluation

Many common words in the test set are also seen in the training set. Rather than artificially boosting the accuracy rate, we will evaluate performance on unique words rather than all words individually.

Some surface forms have more than one possible root form. For example, the word *purōn* may be inflected from the noun *purá* (“altar”), or *purós* (“wheat”), or *pūr* (“fire”). It would be necessary to examine the context to select the appropriate noun, but morphological disambiguation (Hakkani-Tür et al., 2002) is beyond the scope of this paper. In these cases, legitimate root forms proposed by our analyzer may be rejected, but we pay this price in return for an automatic evaluation procedure.

5 Nearest-Neighbor Approach

The memory-based machine learning framework performs well on a benchmark of language learning tasks (Daelemans, 1999), including morphological segmentation of Dutch (van den Bosch, 1999). In this framework, feature vectors are extracted from the training set and stored in a database of instances, called the *instance base*. A distance metric is then defined. For each test instance, its set of nearest neighbors is retrieved from the instance base, and the majority label of the set is returned.

We now adapt this framework to our task, first defining the distance metric (current section), then describing the search algorithm for nearest neighbors (§6).

5.1 Distance Metric

Every word consists of a stem, a (possibly empty) prefix and a (possibly empty) suffix. If two words share a common stem, one can be transformed to the other by substituting its prefix and suffix with their counterparts in the other word. We will call these substitutions the *prefix transformation* and the *suffix transformation*.

The “distance” between two words is to be defined in terms of these transformations. It would be desirable for words that are inflected from the same root to be near neighbors. A distance metric can achieve this effect by favoring prefix and suffix transformations that are frequently observed among words inflected from the same root. We thus provisionally define “distance” as the sum of the frequency counts of the prefix and suffix transformations required to turn one word to the other.

5.2 Stems and Affixes

Defining “Stem” To count the frequencies of prefix and suffix transformations, the stem of each word in the training set must be determined. Ideally, all words inflected from the same root should share the same stem. Unfortunately, for ancient Greek, it is difficult to insist upon such a common stem. In some cases, the stems are completely different⁵; in others, the common stem is obfuscated

⁵Each verb can have up to six different stems, known as the “principal parts”. In extreme cases, a stem may appear completely unrelated to the root on the surface. For example, *oíso* and *énegkon* are both stems of the root *phéro* (“to carry”). A comparable example in English is the inflected verb form *went* and its root form *go*.

Word	Prefix	Stem	Suffix	Transformation	
				Prefix Transformation	Suffix Transformation
(root) <i>lúo</i>	-	<i>lú</i>	<i>o</i>	(root,1)	$\epsilon \leftrightarrow e$ $\underline{o} \leftrightarrow \text{eto}$
(1) <i>elúeto</i>	<i>e</i>	<i>lú</i>	<i>eto</i>	(root,2)	$\epsilon \leftrightarrow \text{para}$ $\underline{o} \leftrightarrow \text{sai}$
(2) <i>paralūsai</i>	<i>para</i>	<i>lū</i>	<i>sai</i>	(root,3)	$\epsilon \leftrightarrow \text{ek}$ $\underline{o} \leftrightarrow \text{thésontai}$
(3) <i>ekluthésontai</i>	<i>ek</i>	<i>lu</i>	<i>thésontai</i>	(1,2)	$e \leftrightarrow \text{para}$ $\text{eto} \leftrightarrow \text{sai}$
				(1,3)	$e \leftrightarrow \text{ek}$ $\text{eto} \leftrightarrow \text{thésontai}$
				(2,3)	$\text{para} \leftrightarrow \text{ek}$ $\text{sai} \leftrightarrow \text{thésontai}$

Table 4: The verb root *lúo* (“to loosen”) and three of its inflected forms are shown. Each inflected form is compared with the root form, as well as the other inflected forms. The “stem”, defined as the longest common substring, is determined for each pair. The prefix and suffix transformations are then extracted. ϵ represents the empty string.

in surface forms due to spelling changes⁶.

We resort to a functional definition of “stem” — the longest common substring of a pair of words. Some examples are shown in Table 4.

Refinements to Definition Three more refinements to the definition of “stem” have been found to be helpful. First, accents are ignored when determining the longest common substring. Accents on stems often change in the process of inflection. These changes are illustrated in Table 4 by the stem *lu*, whose letter *u* has an acute accent, a circumflex accent, and no accent in the three inflected forms.

Second, a minimum length is required for the stem. On the one hand, some pairs, such as *ágo* (“to lead”) and *áxo*, do have a stem of length one (“*a*”). On the other hand, allowing very short stems can hurt performance, since many spurious stems may be misconstrued, such as “*e*” between *phéro* and *énegkon*. The minimum stem length is empirically set at two for this paper.

Length alone cannot filter out all spurious stems. For example, for the pair *patéo* (“to walk”) and an inflected form *katépátesan*, there are two equally long candidate stems, **ate* and *pat*. The latter yields affixes such as “*-éo*” and “*-esan*”, which are relatively frequent⁷. On this basis, the latter stem is chosen.

Some further ways to reduce the noise are to require an affix transformation to occur at least a minimum number of times in the training set, and to restrict the phonological context in which

the transformation can be applied⁸. While significantly reducing recall, these additional restrictions yield only a limited boost in precision.

6 Algorithm

In the training step, a set of prefix and suffix transformations, along with their counts, is compiled for each part-of-speech. These counts enable us to compute the distance between any two words, and hence determine the “nearest neighbor” of a word.

At testing, given an inflected form, its neighbor is any word to which it can be transformed using the affix transformations. We first try to find its nearest neighbor in the training set (§6.1); if no neighbor is found, a novel root is predicted (§6.2).

6.1 Finding Known Roots

If the input word itself appears in the training set, we simply look up its morphological analysis.

If the input word is not seen in the training set, its root form or another inflected form may still be found. We try to transform the input word to the nearest such word, i.e., by using the most frequent prefix and suffix transformations, according to the distance metric (§5.1).

Irregular Stem Spelling Typically, if there are no spelling changes in the stem, the input word can be transformed directly to the root, e.g., from *phéreis* to *phéro*. If the spelling of the stem is substantially different, it is likely to be transformed to another inflected form of the root that contains the same irregular stem. For example, the word *prosexénegken* bears little resemblance to its root *phéro*, but it can be mapped to the word *énegken*

⁶For example, the stem *oz* in the root form *ózo* (“to smell”) is changed to *os* in *exósthesan*, an aorist passive form.

⁷The frequency of each affix is counted in a preliminary round, with each affix receiving a half count in cases of tied stem length.

⁸For example, a certain suffix transformation may be valid only when the stem ends in certain letters.

in the training set, from which we retrieve its root form *phé̄ro*.

Search Order Some affixes are circumfixes; that is, both the prefix and the suffix must occur together. For example, the suffix *-eto* cannot be applied on its own, but must always be used in conjunction with the prefix *e-*, to form words such as *elúeto*, as shown in Table 4.

Other affixes, however, can freely mix with one another, and not all combinations are attested in the training set. This is particularly common when the prefix contains two or more prepositions. For example, the combination *dia-kata-* occurs only two times in the training set, but it can potentially pair with a large number of different suffixes.

Hence, the search for neighbors proceeds in two stages. In the first stage (denoted CIRCUMFIX), the search is restricted to circumfixes, that is, requiring that at least one word-pair in the training set contain both the prefix and suffix transformations. This restriction is prone to data sparseness; if no neighbor is found, the prefix and suffix transformations are then allowed to be applied separately in the second stage (denoted PREFIX/SUFFIX).

6.2 Proposing Novel Roots

A word may be derived from a root of which no inflected form is seen in the training set. Naturally, no neighbor would be found in the previous step, and a novel root must be proposed. We apply the prefix and suffix transformations learned in §5.2, using only circumfixes observed between an inflected form and a root form. For obvious reasons, the resulting string is no longer required to be a neighbor, i.e., a word seen in the training set.

Typically, the various transformations produce many candidate roots. For example, the word *homometríou* (“born of the same mother”), a masculine genitive adjective, can be transformed to its root adjective *homomé̄trios*, but it could equally well be transformed into a hypothetical neuter noun, **homomé̄trion*. Both are perfectly plausible roots.

The automatically discovered affix transformations inevitably contain some noise. When dealing with known roots, much of the noise is suppressed because misapplications of these transformations seldom turn the input word into a real word found in the training set. When proposing novel roots, we no longer enjoy this constraint. Although the

distance metric still helps discriminate against invalid candidates, the increased ambiguity leads to lower accuracy. We address this issue by exploiting a large, unlabelled corpus.

Use of Unlabelled Corpus If a proposed root form is correct, it should be able to generate some inflected forms attested in a large corpus. Intuitively, the “productivity” of the root form may correlate with its correctness.

To generate inflected forms from a root, we simply take the set of affix transformations observed from inflected forms to roots, and reverse the transformations. Continuing with the above example, we generate inflected forms for both candidate roots, the adjective *homomé̄trios*, and the hypothetical neuter noun **homomé̄trion*. While a few inflected forms are generated by both candidates, three are unique to the adjective — *homomé̄trios*, *homomé̄trioi* and *homomé̄trian* — the nominative masculine singular and plural, and the accusative feminine singular, respectively. None of these could have been inflected from a neuter noun.

A straightforward notion of “productivity” of a root would be simply the number of inflected forms attested in the large corpus. It can be further refined, however, by considering the prevalence of the inflected forms. That is, a form generated with more common affix transformations should be given greater weight than one generated with less common ones. Suppose two candidate roots, the adjective *telesphóros* (“bringing to an end”) and the hypothetical verb **telesphoró̄o*, are being considered. Both can generate the inflected form *telesphórou*, the former as the masculine genitive adjective, and the latter as either an imperfect indicative or present imperative contract verb. Since the inflection of the adjective is more frequent in the training set than that of the relatively rare class of contract verbs, the existence of *telesphórou* should lend greater weight to the adjective.

Hence, the “productivity” metric of a novel root is the number of words in the large corpus that it can generate with affix transformations, weighted by the frequencies of those transformations.

7 Experiments

Some statistics on the test set are presented in Table 3. Of the 7,381 words that are seen in the training set, 98.2% received the correct root form. The

Transformation Type	Proportion	Accuracy
CIRCUMFIX	77.5%	94.5%
PREFIX/SUFFIX	10.8%	61.2%
Novel Roots	11.7%	50.0%
Overall	100%	85.7%

Table 5: After excluding known words, which attain an accuracy of 98.2%, the performance on the remaining 3437 unique words in the test set is shown above. Please see §7 for discussions. Results for novel roots are presented in further detail in Table 6.

remaining 1.8% had multiple possible roots; an examination of the context would be needed for disambiguation (see comments in §4.3).

Table 5 presents the accuracy of the predicted roots, after excluding the 7,381 seen words. The result is broken down according to the type of transformation; for the “Novel Roots” type, more detailed results are presented in Table 6.

As discussed in §6.1, the algorithm first searched with CIRCUMFIX. For 77.5% of the words, a neighbor was found using this subset of affix transformations. The rest were then processed using the back-up procedure, PREFIX/SUFFIX, allowing prefix and suffix transformations culled from different word-pairs. This procedure found neighbors for 10.8% of the words; novel roots were hypothesized for the remainder.

Not surprisingly, known roots were more reliably predicted (94.5%) with circumfixes than with separate prefixes and suffixes (61.2%), but both categories still achieved higher accuracy than the challenging task of proposing novel roots (50.0%). We now take a closer look at the errors for both known and novel roots.

7.1 Known Roots

There are three main sources of error. The first is noise in the affix transformations. For example, the spurious prefix transformation $p \leftrightarrow ph$ was derived from the pair *phérō* and *perienégkasan*. When applied on *pasátō*, along with a suffix transformation, it yielded the false root form *phásko*.

A second source can be attributed to incorrect affix boundaries. For example, *ekteínantes* was misconstrued as having “*e-*” rather than the preposition *ek* as prefix. This prefix is by itself perfectly viable, but “*e-*” and “*-antes*” cannot occur together as a circumfix. The resulting string hap-

Evaluation Method	Accuracy
BASELINE	45.0%
TLG RERANK	50.0%
+ <i>Ignore accents</i>	55.2%
+ <i>Oracle POS</i>	65.5%

Table 6: Results for predicting novel roots, for the 402 words for whom no neighbor was found. BASELINE uses the distance metric (§5.1) as before; TLG RERANK exploits the unlabelled Thesaurus Linguae Graecae corpus to re-rank the top candidates (§6.2) proposed by BASELINE.

pened to match the root *kteíno*, rather than the true root *teíno*.

A third source is confusion between parts-of-speech, most commonly noun and verb. For example, the nearest neighbor of the genitive noun *lupōn* was the verb *lupései*, yielding the verb root *lupéo* rather than the noun *lúpe*.

7.2 Novel Roots

As a baseline, the distance metric (§5.1) was used alone to rank the novel candidate roots. As seen in Table 6, performance dropped to 45.0%.

When the Thesaurus Linguae Graecae corpus was utilized to rerank the novel candidate roots proposed by the baseline, an absolute gain⁹ of 5% was achieved. A further 5.2% of the mistakes were due to placing the accent incorrectly, such as *ktenótrophos* rather than *ktenotróphos*, mostly on nouns and adjectives. These mistakes are difficult to rectify, since multiple positions are often possible¹⁰.

Finally, to measure the extent to which part-of-speech (POS) confusions are responsible, we performed an experiment in which the gold-standard POS of each word was supplied to the analyzer (see “Oracle POS” in Table 6). When deriving novel roots, only those affix transformations belonging to the oracle POS were considered. With this constraint, accuracy rose to 65.5%.

⁹The significance level is at $p = 0.11$, according to McNemar’s test. The improvement is not statistically significant, and may be a reflection of the relatively small test set.

¹⁰The accent in an inflected noun retains its position in the root, unless that position violates certain phonological rules. In many cases, there is no reliable way to predict the accent position in the root noun from the position in the inflected form.

8 Conclusion

We have proposed a nearest-neighbor machine learning framework for analyzing ancient Greek morphology. This framework is data-driven, with automatic discovery of stems and affixes. The analyzer is able to predict novel roots. A significant novelty is the exploitation of a large, unlabelled corpus to improve performance.

We plan to further improve the derivation of novel roots by predicting their parts-of-speech from context, and by incorporating distributional information (Yarowsky and Wicentowski, 2000).

Acknowledgments

The author would like to thank Stephanie Seneff, Kalliroi Georgila, Konstantinos Katsiapis and Steven Lulich for their insightful comments.

References

- Meni Adler, Yoav Goldberg, David Gabay, and Michael Elhadad. 2008. Unsupervised Lexicon-based Resolution of Unknown Words for Full Morphological Analysis. *Proc. ACL*. Columbus, OH.
- Luci Berkowitz and Karl A. Squitter. 1986. *Thesaurus Linguae Graecae*. Oxford University Press, UK.
- Antal van den Bosch and Walter Daelemans. 1999. Memory-based Morphological Analysis. *Proc. ACL*. College Park, MD.
- Gregory Crane. 1991. Generating and Parsing Classical Greek. *Literary and Linguistic Computing*, 6(4):243–245.
- Gregory Crane. 1996. *Perseus 2.0: Interactive Sources and Studies on Ancient Greece*. Yale University Press, New Haven, CT.
- Walter Daelemans, Antal van den Bosch and Jakub Zavrel. 1999. Forgetting Exceptions is Harmful in Language Learning. *Machine Learning*, 34:11–41.
- Sajib Dasgupta and Vincent Ng. 2007. High-Performance, Language-Independent Morphological Segmentation. *Proc. HLT-NAACL*. Rochester, NY.
- John Goldsmith. 2001. Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics*, 27(2):153–198.
- Dilek Z. Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2002. Statistical Morphological Disambiguation for Agglutinative Languages. *Computers and the Humanities*, 36(4):381–410.
- Samarth Keshava and Emily Pitler. 2006. A Simpler, Intuitive Approach to Morpheme Induction. *Proc. 2nd PASCAL Challenges Workshop*. Venice, Italy.
- Kimmo Koskenniemi. 1983. Two-level morphology: a general computation model for word-form recognition and production. *Publication No. 11, Department of General Linguistics, University of Helsinki*. Helsinki, Finland.
- Krister Lindén. 2008. A Probabilistic Model for Guessing Base Forms of New Words by Analogy. *Proc. CICLing*. Haifa, Israel.
- David W. Packard. 1973. Computer-assisted Morphological Analysis of Ancient Greek. *Proc. 5th Conference on Computational Linguistics*. Pisa, Italy.
- David Yarowsky and Richard Wicentowski. 2000. Minimally Supervised Morphological Analysis by Multimodal Alignment. *Proc. ACL*. Hong Kong, China.

Context-based Arabic Morphological Analysis for Machine Translation

ThuyLinh Nguyen

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
thuylinh@cs.cmu.edu

Stephan Vogel

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
vogel@cs.cmu.edu

Abstract

In this paper, we present a novel morphology preprocessing technique for Arabic-English translation. We exploit the Arabic morphology-English alignment to learn a model removing nonaligned Arabic morphemes. The model is an instance of the Conditional Random Field (Lafferty et al., 2001) model; it deletes a morpheme based on the morpheme's context. We achieved around two BLEU points improvement over the original Arabic translation for both a travel-domain system trained on 20K sentence pairs and a news domain system trained on 177K sentence pairs, and showed a potential improvement for a large-scale SMT system trained on 5 million sentence pairs.

1 Introduction

Statistical machine translation (SMT) relies heavily on the word alignment model of the source and the target language. However, there is a mismatch between a rich morphology language (e.g Arabic, Czech) and a poor morphology language (e.g English). An Arabic source word often corresponds to several English words. Previous research has focused on attempting to apply morphological analysis to machine translation in order to reduce unknown words of highly inflected languages. Nießen and Ney (2004) represented a word as a vector of morphemes and gained improvement over word-based system for

German-English translation. Goldwater and McClosky (2005) improved Czech-English translation by applying different heuristics to increase the equivalence of Czech and English text.

Specially for Arabic-English translation, Lee (2004) used the Arabic part of speech and English parts of speech (POS) alignment probabilities to retain an Arabic affix, drop it from the corpus or merge it back to a stem. The resulting system outperformed the original Arabic system trained on 3.3 million sentence pairs corpora when using monotone decoding. However, an improvement in monotone decoding is no guarantee for an improvement over the best baseline achievable with full word forms. Our experiments showed that an SMT phrase-based translation using 4 words distance reordering could gain four BLEU points over monotone decoding. Sadat and Habash (2006) explored a wide range of Arabic word-level preprocessing and produced better translation results for a system trained on 5 million Arabic words.

What all the above methodologies do not provide is a means to disambiguate morphological analysis for machine translation based on the words' contexts. That is, for an Arabic word analysis of the form *prefix*-stem-suffix** a morpheme only is either always retained, always dropped off or always merged to the stem regardless of its surrounding text. In the example in Figure (1), the Arabic word “AlnAfi*h” (“window” in English) was segmented as “Al nAfi* ap”. The morpheme “ap” is removed so that “Al nAfi*” aligned to “the window” of the English sentence. In the sentence “hl ldyk mqAEd bjwAr AlnAf*h ?” (“do you have window tables ?” in English) the word “AlnAfi*h” is also segmented as “Al nAfi* ap”. But in this sentence, morphological preprocessing should remove both “Al” and “ap” so that only the remain-

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

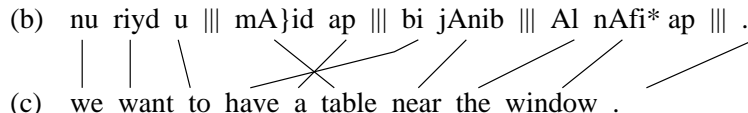
- (a) nryd mA}dh bjAnb AlnAf*h .
- (b) nu riyd u ||| mA}id ap ||| bi jAnib ||| Al nAfi* ap ||| .

- (c) we want to have a table near the window .
- (d) nu riyd u mA}id ap bi jAnib Al nAfi* .

Figure 1: (a) Romanization of original Arabic sentence, (b) Output of morphological analysis toolkit—words are separated by ‘|||’, (c) English translation and its alignment with full morphological analysis (d) Morphological analysis after removing unaligned morphemes.

ing morpheme “nAfi*” aligned to the word “window” of the English translation. Thus an appropriate preprocessing technique should be guided by English translation and bring the word context into account.

In this paper we describe a context-based morphological analysis for Arabic-English translation that take full account morphemes alignment to English text. The preprocessing uses the Arabic morphology disambiguation in (Smith et al., 2005) for full morphological analysis and learns the removing morphemes model based on the Viterbi alignment of English to full morphological analysis. We tested the model with two training corpora of 5.2 millions Arabic words(177K sentences) in news domain and 159K Arabic words (20K sentences) in travel conversation domain and gain improvement over the original Arabic translation in both experiments. The system that trained on a subsample corpora of 5 millions sentence pairs corpora also showed one BLEU score improvement over the original Arabic system on unseen test set.

We will explain our technique in the next section and briefly review the phrase based SMT model in section 3. The experiment results will be presented in section 4.

2 Methodology

We first preprocess the Arabic training corpus and segment words into morpheme sequences of the form *prefix* stem suffix**. Stems are verbs, adjectives, nouns, pronouns, etc., carrying the content of the sentence. Prefixes and suffixes are functional morphemes such as gender and case markers, prepositions, etc. Because case makers do not exist in English, we remove case marker suffixes from the morphology output. The output of this process is a *full morphological analysis* corpus. Even after removing case markers, the token count

of the full morphology corpus still doubles the original Arabic’s word token count and is approximately 1.7 times the number of tokens of the English corpus. As stated above, using original Arabic for translation introduces more unknown words in test data and causes multiple English words to map to one Arabic word. At the morpheme level, an English word would correspond to a morpheme in the full morphology corpus but some prefixes and suffixes in the full morphology corpus may not be aligned with any English words at all. For example, the Arabic article “Al” (“the” in English) prefixes to both adjectives and nouns, while English has only one determiner in a simple noun phrase. Using the full morphological analysis corpus for translation would introduce redundant morphemes in the source side.

The goal of our morphological analysis method for machine translation is *removing nonaligned* prefixes and suffixes from the full morphology corpus using a data-driven approach. We use the word alignment output of the full morphology corpus to the English corpus to delete morphemes in a sentence. If an affix is not aligned to an English word in the word alignment output, the affix should be removed from the morphology corpus for better one-to-one alignment of source and target corpora. However, given an unseen test sentence, the English translation of the sentence is not available to remove affixes based on the word alignment output. We therefore learn a model removing non-aligned morphemes from the full morphology Arabic training corpus and its alignment to the English corpus. To obtain consistency between training corpus and test set, we applied the model to both Arabic training corpus and test set, obtaining preprocessed morphology corpora for the translation task.

In this section, we will explain in detail each steps of our preprocessing methodology:

- Apply word segmentation to the Arabic training corpus to get the full morphological analysis corpus.
- Annotate the full morphological analysis corpus based on its word alignment to the English training corpus. We tag a morpheme as “Deleted” if it should be removed from the corpus, and “Retained” otherwise.
- Learn the morphology tagger model.
- Apply the model to both Arabic training corpus and Arabic test corpus to get preprocessed corpus for translation.

2.1 Arabic Word Segmentation

Smith et al. (2005) applies a source-channel model to the problem of morphology disambiguation. The source model is a uniform model that defines the set of analyses. For Arabic morphology disambiguation, the source model uses the list of un-weighted word analyses generated by BAMA toolkit (Buckwalter, 2004). The channel model disambiguates the morphology alternatives. It is a log-linear combination of features, which capture the morphemes’ context including tri-gram morpheme histories, tri-gram part-of-speech histories and combinations of the two.

The BAMA toolkit and hence (Smith et al., 2005) do not specify if a morpheme is an affix or a stem in the output. Given a segmentation of an original Arabic word, we considered a morpheme a_i as a stem if its parts of speech p_i is either a noun, pronoun, verb, adjective, question, punctuation, number or abbreviation. A morpheme on the left of its word’s stem is a prefix and it is a suffix if otherwise. We removed case marker morphemes and got the full morphology corpus.

2.2 Annotate Morphemes

To extract the Arabic morphemes that align to English text, we use English as the source corpus and aligned to Arabic morpheme corpus using GIZA++ (Och and Ney, 2003) toolkit. The IBM3 and IBM4 (Brown et al., 1994) word alignment models select each word in the source sentence, generate fertility and a list of target words that connect to it. This generative process would constrain source words to find alignments in the target sentence. Using English as source corpus, the alignment models force English words to generate their alignments in the Arabic morphemes.

GIZA++ outputs Viterbi alignment for every sentence pair in the training corpus as depicted in (b) and (c) of Figure (1). In our experiment, only 5% of English words are not aligned to any Arabic morpheme in the Viterbi alignment. From Viterbi English-morpheme alignment output, we annotate morphemes either to be deleted or retained as follows:

- Annotate stem morphemes as “Retained”(R), in dependant of word alignment output.
- Annotate a prefix or a suffix as “Retained” (R) if it is aligned to an English word.
- Annotate a prefix or a suffix as “Deleted” (D) if it is not aligned to an English word.

Note that the model does not assume that GIZA++ outputs accurate word alignments. We lessen the impact of the GIZA++ errors by only using the word alignment output of prefix and suffix morphemes.

Furthermore, because the full morphology sentence is longer, each English word could align to a separate morpheme. Our procedure of annotating morphemes also constrains morphemes tagged as “Retained” to be aligned to English words. Thus if we remove “Deleted” morphemes from the morphology corpus, the reduced corpus and English corpus have the property of one-to-one mapping we prefer for source-target corpora in machine translation.

2.3 Reduced Morphology Model

The reduced morphology corpus would be the best choice of morphological analysis for machine translation. Because it is impossible to tag morphemes of a test sentence without the English reference based on Viterbi word alignment, we need to learn a morpheme tagging model. The model estimates the distributions of tagging sequences given a morphologically analysed sentence using the previous step’s annotated training data.

The task of tagging morphemes to be either “Deleted” or “Retained” belongs to the set of sequence labelling problems. The conditional random fields (CRF) (Lafferty et al., 2001) model has shown great benefits in similar applications of natural language processing such as part-of-speech tagging, noun phrase chunking (Sha and Pereira, 2003), morphology disambiguation(Smith et al., 2005). We apply the CRF model to our morpheme tagging problem.

Let $\mathcal{A} = \{(\mathbf{A}, \mathbf{T})\}$ be the full morphology training corpus where $\mathbf{A} = a_1|p_1 a_2|p_2 \dots a_m|p_m$ is a morphology Arabic sentence, a_i is a morpheme in the sentence and p_i is its POS; $\mathbf{T} = t_1 t_2 \dots t_m$ is the tag sequence of \mathbf{A} , each t_i is either ‘‘Deleted’’ or ‘‘Retained’’. The CRF model estimates parameter $\bar{\theta}^*$ maximizing the conditional probability of the sequences of tags given the observed data:

$$\bar{\theta}^* = \underset{\bar{\theta}}{\operatorname{argmax}} \sum_{(\mathbf{A}, \mathbf{T}) \in \mathcal{A}} \tilde{p}((\mathbf{A}, \mathbf{T})) \log p(\mathbf{T}|\mathbf{A}, \bar{\theta}) \quad (1)$$

where $\tilde{p}((\mathbf{A}, \mathbf{T}))$ is the empirical distribution of the sentence (\mathbf{A}, \mathbf{T}) in the training data, $\bar{\theta}$ are the model parameters. The model’s log conditional probability $\log p(\mathbf{T}|\mathbf{A}, \bar{\theta})$ is the linear combination of feature weights:

$$\log p(\mathbf{T}|\mathbf{A}, \bar{\theta}) = \sum_k \theta_k f_k((\mathbf{A}_q, \mathbf{T}_q)) \quad (2)$$

The feature functions $\{f_k\}$ are defined on any subset of the sentence $\mathbf{A}_q \subset \mathbf{A}$ and $\mathbf{T}_q \subset \mathbf{T}$. CRFs can accommodate many closely related features of the input. In our morpheme tagging model, we use morpheme features, part-of-speech features and combinations of both. The features capture the local contexts of morphemes. The lexical morpheme features are the combinations of the current morpheme and up to 2 previous and 2 following morphemes. The part-of-speech features are the combinations of the current part of speech and up to 3 previous part of speeches. The part of speech, morpheme combination features capture the dependencies of current morphemes and up to its 3 previous parts of speech.

2.4 Preprocessed Data

Given a full morphology sentence \mathbf{A} , we use the morpheme tagging model learnt as described in the previous section to *decode* \mathbf{A} into the most probable sequence of tags $\mathbf{T}^* = t_1 t_2 \dots t_m$.

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmax}} \Pr(\mathbf{T}|\mathbf{A}, \bar{\theta}^*) \quad (3)$$

If a t_i is ‘‘Deleted’’, the morpheme a_i is removed from the morphology sentence \mathbf{A} . The same procedure is applied to both training Arabic corpus and test corpus to get preprocessed data for translation. We call a morphology sentence after removing ‘‘Deleted’’ tag a *reduced* morphology sentence.

In our experiments, we used the freely available CRF++¹ toolkit to train and decode with the morpheme tagging model. The CRF model smoothed the parameters by assigning them Gaussian prior distributions.

3 Phrase-based SMT System

We used the open source Moses (Koehn, 2007) phrase-based MT system to test the impact of the preprocessing technique on translation results. We kept the default parameter settings of Moses for translation model generation. The system used the ‘‘grow-diag-final’’ alignment combination heuristic. The phrase table consisted of phrase pairs up to seven words long. The system used a tri-gram language model built from SRI (Stolcke, 2002) toolkit with modified Kneser-Ney interpolation smoothing technique (Chen and Goodman, 1996). By default, the Moses decoder uses 6 tokens distance re-ordering windows.

4 Experiment Results

In this section we present experiment results using our Arabic morphology preprocessing technique.

4.1 Data Sets

We tested our morphology technique on a small data set of 20K sentence pairs and a medium size data set of 177K sentence pairs.

4.1.1 BTEC Data

As small training data set we used the BTEC corpus (Takezawa et al., 2002) distributed by the International Workshop on Spoken Language Translation (IWSLT) (Eck and Hori, 2005). The corpus is a collection of conversation transcripts from the travel domain. Table 1 gives some de-

	Arabic			Eng
	Ori	Full	Reduced	
Sentences	19972			
Tokens	159K	258K	183K	183K
Types	17084	8207	8207	7298

Table 1: BTEC corpus statistics

tails for this corpus, which consists of nearly 20K sentence pairs with lower case on the English side. There is an imbalance of word types and word tokens between original Arabic and English. The

¹<http://crfpp.sourceforge.net/>

original Arabic sentences are on average shorter than the English sentences whereas the Arabic vocabulary is more than twice the size of the English vocabulary. The word segmentation reduced the number of word types in the corpus to be closed to English side but also increased word tokens quite substantially. By removing nonaligned morphemes, the reduced corpus is well balanced with the English corpus.

The BTEC experiments used the 2004 IWSLT Evaluation Test set as development set and 2005 IWSLT Evaluation Test set as unseen test data. Table 2 gives the details of the two test sets. Both of them had 16 reference translations per source sentence. The English side of the training corpus was used to build the language model. To optimize the parameters of the decoder, we performed minimum error rate training on IWSLT04 optimizing for the IBM-BLEU metric (Papineni et al., 2002).

4.1.2 Newswire Corpora

We also tested the impact of our morphology technique on parallel corpus in the news domain. The corpora were collected from LDC’s full Arabic news translation corpora and a small portion of UN data. The details of the data are give in Table 3. The data consists of 177K sentence pairs, 5.2M words on the Arabic and 6M words on the English side.

	Arabic			Eng
	Ori	Full	Reduced	
Sentences	177035			
Tokens	5.2M	9.3M	6.2M	6.2M
Types	155K	47K	47K	68K

Table 3: Newswire corpus statistics

We used two test sets from past NIST evaluations as test data. NIST MT03 was used as development set for optimizing parameters with respect to the IBM-BLEU metric, NIST MT06 was used as unseen test set. Both test sets have 4 references per test sentence. Table 4 describes the data statistics of the two test sets. All Newswire translation experiments used the same language model estimated from 200 million words collected from the Xinhua section of the GIGA word corpus.

4.2 Translation Results

4.2.1 BTEC

We evaluated the machine translation according to the case-insensitive BLEU metric. Table 5 shows the BTEC results when translated with default Moses setting of distance-based reordering window size 6. The original Arabic word translation was the baseline of the evaluation. The second row contains translation scores using the full morphology translation. Our new technique of context-based morphological analysis is shown in the last row.

	IWSLT04	IWSLT05
Ori	58.20	54.50
Full	58.55	55.87
Reduced	60.28	56.03

Table 5: BTEC translations results on IBM-BLEU metrics(Case insensitive and 6 tokens distance reordering window). The boldface marks scores significantly higher than the original Arabic translation scores.

The full morphology translation performed similar to the baseline on the development set but outperformed the baseline on the unseen test set. The reduced corpus showed significant improvements over the baseline on the development set (IWSLT04) and gave an additional small improvement over the full morphology score over the unseen data (IWSLT05).

So why did the reduced morphology translation not outperform more significantly the full morphology translation on unseen set IWSLT05? To analysis this in more detail, we selected good full morphology translations and compared them with the corresponding reduced morphology translations. Figure 2 shows one of these examples. Typically, the reduced morphology translations

Source: *لوقت ام جفا ال يننا ينفسوي*
 Ref: *i'm afraid what you are saying doesn't make any sense to me.*
 Full: *i'm sorry i don't understand what you are saying.*
 Reduced: *i'm sorry i don't understand what you say.*

Figure 2: An example of BTEC translation output.

are shorter than both the references and the full morphology outputs. Table 2 shows that for the IWSLT05 test set, the ratio of the average English reference sentence length and the source sen-

	IWSLT04 (Dev set)				IWSLT05 (Unseen set)			
	Arabic			English	Arabic			English
	Ori	Full	Reduced		Ori	Full	Reduced	
Sentences	500			8000	506			8096
Words	3261	5243	3732	64896	3253	5155	3713	66286
Avg Sent Length	6.52	10.48	7.46	8.11	6.43	10.19	7.34	8.18

Table 2: BTEC test set statistics

	MT03 (Dev set)				MT06 (Unseen set)			
	Arabic			English	Arabic			English
	Ori	Full	Reduced		Ori	Full	Reduced	
Sentences	663			2652	1797			7188
Words	16268	27888	18888	79163	41059	71497	48716	222750
Avg Sent Length	24.53	42.06	28.49	29.85	22.85	39.79	27.1	30.98

Table 4: Newswire test set statistics

tence length is slightly higher than the corresponding ratio for IWSLT04. Using the parameters optimised for IWSLT04 to translate IWSLT05 sentences would generate hypotheses slightly shorter than the IWSLT05 references resulting in brevity penalties in the BLEU metric. The IWSLT05 brevity penalties for original Arabic, reduced morphology and full morphology are 0.969, 0.978 and 0.988 respectively. Note that the BTEC corpus and test sets are in the travel conversation domain, the English reference sentences contain a large number of high frequency words. The full morphological analysis with additional prefixes and suffixes outputs longer translations containing high frequency words resulting in a high n-gram match and lower BLEU brevity penalty. The reduced translation method could generate translations that are comparable but do not have the same effect on BLEU metrics.

4.2.2 Newswire results

Table 6 presents the translation results for the Newswire corpus. Even though morphology segmentation reduced the number of unseen words, the translation results of full morphological analysis are slightly lower than the original Arabic scores in both development set MT03 and unseen test set MT06. This is consistent with the result achieved in previous literature (Sadat and Habash, 2006). Morphology preprocessing only helps with small corpora, but the advantage decreases for larger data sets.

Our context dependent preprocessing technique

	MT03	MT06
Ori	45.55	32.09
Full	45.30	31.54
Reduced	47.69	34.13

Table 6: Newswire translation results on IBM-BLEU metrics (Case insensitive and 6 tokens distance reordering window). The boldface marks scores significantly higher than the original Arabic translation’s scores.

shows significant improvements on both development and unseen test sets. Moreover, while the advantage of morphology segmentation diminishes for the full morphology translation, we achieve an improvement of more than two BLEU points over the original Arabic translations in both development set and unseen test set.

4.3 Unknown Words Reduction

A clear advantage of using morphology based translation over original word translation is the reduction in the number of untranslated words. Table 7 compares the number of unknown Arabic tokens for original Arabic translation and reduced morphology translation. In all the test sets, morphology translations reduced the number of unknown tokens by more than a factor of two.

4.4 The Impact of Reordering Distance Limit

The reordering window length is determined based on the movements of the source phrases. On an average, an original Arabic word has two mor-

Reorder Window		0	2	3	4	5	6	7	8	9
IWSLT04	Ori	57.21	57.92	58.01	58.31	58.16	58.20	58.20	58.12	58.01
	Full	56.89	57.54	58.62	58.39	58.32	58.55	58.55	58.55	58.57
	Reduced	58.36	59.56	60.05	60.70	60.32	60.28	60.46	60.30	60.55
MT03	Ori	41.75	43.84	45.24	45.61	45.40	45.55	45.21	45.22	45.19
	Full		41.45	43.12	44.32	44.71	45.30	45.80	45.88	45.82
	Reduced	44.08	45.28	46.50	47.40	47.41	47.69	47.59	47.75	47.79

Table 8: The impact of reordering limits on BTEC’s development set IWSLT04 and Newswire’s development set MT03. The translation scores are IBM-BLEU metric

Test Set	Ori	Reduced
IWSLT04	242	100
IWSLT05	219	97
MT03	1463	553
MT06	3734	1342

Table 7: Unknown tokens count

phemes. The full morphology translation with a 6-word reordering window has the same impact as a 3-word reordering when translating the original Arabic. To fully benefit from word reordering, the full morphology translation requires a longer reorder distance limit. However, in current phrase based translations, reordering models are not strong enough to guide long distance source-word movements. This shows an additional advantage of the nonaligned morpheme removal technique.

We carried out experiments from monotone decoding up to 9 word distance reordering limit for the two development sets IWSLT04 and MT03. The results are given in Table 8. The BTEC data set does not benefit from a larger reordering window. Using only a 2-word reordering window the score of the original Arabic translations(57.92) was comparable to the best score (58.31) obtained by using a 4-word reordering window. On the other hand, the reordering limit showed a significant impact on Newswire data. The MT03 original Arabic translation using a 4-word re-ordering window resulted in an improvement of 4 BLEU points over monotone decoding. Large Arabic corpora usually contain data from the news domain. The decoder might not effectively reorder very long distance morphemes for these data sets. This explains why machine translation does not benefit from word-based morphological segmentation for large data sets which adequately cover the vocabu-

lary of the test set.

4.5 Large Training Corpora Results

We wanted to test the impact of our preprocessing technique on a system trained on 5 million sentence pairs (128 million Arabic words). Unfortunately, the CRF++ toolkit exceeded memory limits when executed even on a 24GB server. We created smaller corpora by sub-sampling the large corpus for the source side of MT03 and MT06 test sets. The sub-sampled corpus have 500K sentence pairs and cover all source phrases of MT03 and MT06 which can be found in the large corpus. In these experiments, we used a lexical reordering model into translation model. The language model was the 5-gram SRI language model built from the whole GIGA word corpus. Table 9

	MT03	MT06
5M Ori	56.22	42.17
Sub-sample Ori	54.54	41.59
Sub-sample Full	51.47	40.84
Sub-sample Reduced	54.78	43.20

Table 9: Translation results of large corpora(Case insensitive, IBM-BLEU metric). The boldface marks score significantly higher than the original Arabic translation score.

presents the translation result of original Arabic system trained on the full 5M sentence pairs corpus and the three systems trained on the 500K sentence pairs sub-sampled corpus. The sub-sampled full morphology system scores degraded for both development set and unseen test set. On development set, the sub-sampled reduced morphology system score was slightly better than baseline. On the unseen test set, it significantly outperformed both the baseline on sub-sampled training data and even outperformed the system trained on the entire

5M sentence pairs.

5 Conclusion and Future Work

In this paper, we presented a context-dependent morphology preprocessing technique for Arabic-English translation. The model significantly outperformed the original Arabic systems on small and mid-size corpora and unseen test set on large training corpora. The model treats morphology processing task as a sequence labelling problem. Therefore, other machine learning techniques such as perceptron (Collins, 2002) could also be applied for this problem.

The paper also discussed the relation between the size of the reordering window and morphology processing. In future investigations, we plan to extend the model such that merging morphemes is included. We also intent to study the impact of phrase length and phrase extraction heuristics.

Acknowledgement

We thank Noah Smith for useful comments and suggestions and providing us with the morphology disambiguation toolkit. We also thank Sameer Badaskar for help on editing the paper. We also thank anonymous reviewers for helpful comments. The research was supported by the GALE project.

References

- Brown, Peter F., Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1994. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.
- Buckwalter, T. 2004. Arabic Morphological Analyzer version 2.0. LDC2004L02.
- Chen, Stanley F. and Joshua Goodman. 1996. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of the ACL*, pages 310–318.
- Collins, Michael. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of EMNLP '02*, pages 1–8.
- Eck, M. and C. Hori. 2005. Overview of the IWSLT 2005 Evaluation Campaign. In *Proceedings of IWSLT*, pages 11–17.
- Goldwater, Sharon and David McClosky. 2005. Improving Statistical MT through Morphological Analysis. In *Proceedings of HLT/EMNLP*, pages 676–683, Vancouver, British Columbia, Canada.
- Koehn, et al. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Annual Meeting of ACL, demonstration session*.
- Lafferty, John, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of 18th ICML*, pages 282–289.
- Lee, Young S. 2004. Morphological Analysis for Statistical Machine Translation. In *HLT-NAACL 2004: Short Papers*, pages 57–60, Boston, Massachusetts, USA.
- Nießen, Sonja and Hermann Ney. 2004. Statistical Machine Translation with Scarce Resources Using Morpho-Syntactic Information. *Computational Linguistics*, 30(2), June.
- Och, Franz Josef and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.
- Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th ACL*, pages 311–318, Philadelphia.
- Sadat, Fatiha and Nizar Habash. 2006. Combination of Arabic Preprocessing Schemes for Statistical Machine Translation. In *Proceedings of the ACL*, pages 1–8, Sydney, Australia. Association for Computational Linguistics.
- Sha, Fei and Fernando Pereira. 2003. Shallow Parsing with Conditional Random Fields. In *Proceedings of NAACL '03*, pages 134–141, Morristown, NJ, USA.
- Smith, Noah A., David A. Smith, and Roy W. Tromble. 2005. Context-Based Morphological Disambiguation with Random Fields. In *Proceedings of HLT/EMNLP*, pages 475–482, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.
- Stolcke, A. 2002. SRILM – an Extensible Language Modeling Toolkit. In *Intl. Conf. on Spoken Language Processing*.
- Takezawa, Toshiyuki, Eiichiro Sumita, Fumiaki Sugaya, Hirofumi Yamamoto, and Seiichi Yamamoto. 2002. Toward a Broad-Coverage Bilingual Corpus for Speech Translation of Travel Conversations in the Real World. In *Proceedings of LREC 2002*, pages 147–152.

A Tree-to-String Phrase-based Model for Statistical Machine Translation

Thai Phuong Nguyen

College of Technology

Vietnam National University, Hanoi

thainp@vnu.edu.vn

Akira Shimazu¹, Tu-Bao Ho², Minh Le Nguyen¹, and Vinh Van Nguyen¹

¹School of Information Science

²School of Knowledge Science

Japan Advanced Institute of Science and Technology

{shimazu,bao,nguyenml,vinhnv}@jaist.ac.jp

Abstract

Though phrase-based SMT has achieved high translation quality, it still lacks of generalization ability to capture word order differences between languages. In this paper we describe a general method for tree-to-string phrase-based SMT. We study how syntactic transformation is incorporated into phrase-based SMT and its effectiveness. We design syntactic transformation models using unlexicalized form of synchronous context-free grammars. These models can be learned from source-parsed bitext. Our system can naturally make use of both constituent and non-constituent phrasal translations in the decoding phase. We considered various levels of syntactic analysis ranging from chunking to full parsing. Our experimental results of English-Japanese and English-Vietnamese translation showed a significant improvement over two baseline phrase-based SMT systems.

1 Introduction

Based on the kind of linguistic information which is made use of, syntactic SMT can be divided into four types: tree-to-string, string-to-tree, tree-to-tree, and hierarchical phrase-based. The tree-to-string approach (Collins et al., 2005; Nguyen and Shimazu, 2006; Liu et al., 2006 and 2007) supposes that syntax of the source language is known. This approach can be applied when a source language parser is available. The string-to-tree approach (Yamada and Knight, 2001; Galley et al., 2006) focuses on syntactic modelling of the target language in cases it has syntactic resources such as treebanks and parsers. The tree-to-tree approach models the syntax of both languages, therefore extra cost is required. The fourth approach (Chiang, 2005) constraints phrases under context-free grammar structure without any requirement of linguistic annotation.

In this paper, we present a tree-to-string phrase-based method which is based on synchronous CFGs. This method has two important properties: syntactic transformation is used in the decoding phase including a word-to-phrase tree transformation model and a phrase reordering model; phrases are the basic unit of translation. Since we design syntactic transformation models using unlexicalized synchronous CFGs, the number of rules is small¹. Previous studies on tree-to-string SMT are different from ours. Collins et al. (2005) used hand crafted rules to carry out word reordering in the preprocessing phase but not decoding phase. Nguyen and Shimazu (2006) presented a more general method in which lexicalized syntactic reordering models based on PCFGs can be learned from source-parsed bitext and then applied in the preprocessing phase. Liu et al. (2006) changed the translation unit from phrases to tree-to-string alignment templates (TATs) while we do not. TATs were represented as xRs rules while we use synchronous CFG rules. In order to overcome the limitation that TATs can not capture non-constituent phrasal translations, Liu et al. (2007) proposed forest-to-string rules while our system can naturally make use of such kind of phrasal translation by word-to-phrase tree transformation.

We carried out experiments with two language pairs English-Japanese and English-Vietnamese. Our system achieved significant improvements over Pharaoh, a state-of-the-art phrase-based SMT system. We also analyzed the dependence of translation quality on the level of syntactic analysis (shallow or deep).

Figure 1 shows the architecture of our system. The input of this system is a source-language tree and the output is a target-language string. This system uses all features of conventional phrase-based SMT as in (Koehn et al., 2003). There are two new features including a word-to-phrase tree transformation model and a phrase reordering model. The decoding algo-

¹See Section 6.2.

rithm is a tree-based search algorithm.

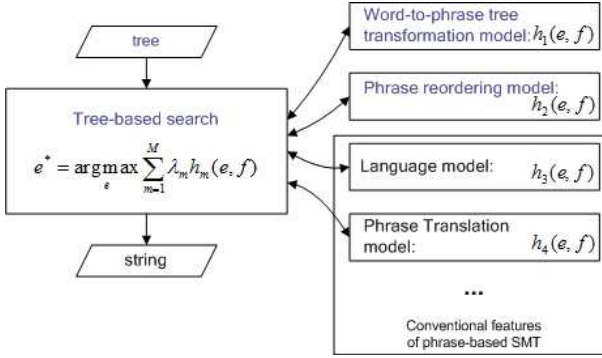


Figure 1: A syntax-directed phrase-based SMT architecture.

2 Translation Model

We use an example of English-Vietnamese translation to demonstrate the translation process as in Figure 2. Now we describe a tree-to-string SMT model based on synchronous CFGs. The translation process is:

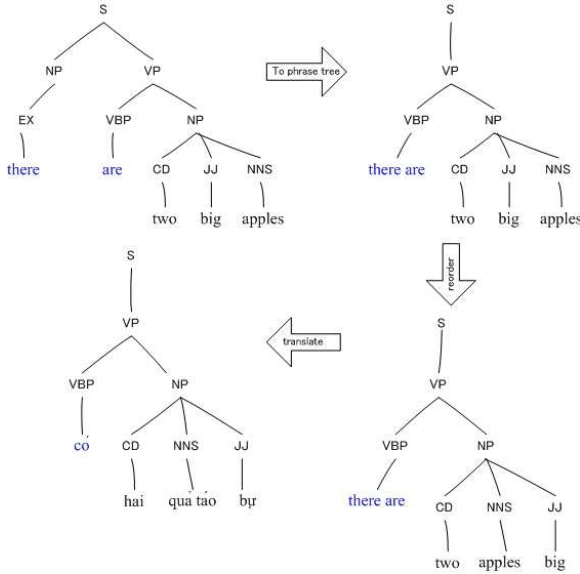


Figure 2: The translation process.

$$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \quad (1)$$

where T_1 is a source tree, T_2 is a source phrase tree, T_3 is a reordered source phrase tree, and T_4 is a target phrase tree.

Using the first order chain rule, the joint probability over variables (trees) in graphical representation 1 is approximately calculated by:

$$P(T_1, T_2, T_3, T_4) = P(T_1) \times P(T_2|T_1) \times P(T_3|T_2) \times P(T_4|T_3) \quad (2)$$

$P(T_1)$ can be omitted since only one syntactic tree is used. $P(T_2|T_1)$ is a word-to-phrase tree transformation model we describe later. $P(T_3|T_2)$ is a reordering model. $P(T_4|T_3)$ can be calculated using a phrase translation model and a language model. This is the fundamental equation of our study represented in this paper. In the next section, we will describe how to transform a word-based CFG tree into a phrase-based CFG tree.

3 Word-to-Phrase Tree Transformation

3.1 Penn Treebank's Tree Structure

According to this formalism, a tree is represented by phrase structure. If we extract a CFG from a tree or set of trees, there will be two possible rule forms:

- $A \rightarrow \alpha$ where α is a sequence of nonterminals (syntactic categories).
- $B \rightarrow \gamma$ where γ is a terminal symbol (or a word in this case).

We consider an example of a syntactic tree and a simple CFG extracted from that tree.

Sentence: "I am a student"
 Syntactic tree: $(S (NP (NN I)) (VP (VBP am) (NP (DT a) (NN student))))$
 Rule set: $S \rightarrow NP VP$; $VP \rightarrow VBP NP$; $NP \rightarrow NN | DT NN$; $NN \rightarrow I | student$; $VBP \rightarrow am$; $DT \rightarrow a$

However, we are considering phrase-based translation. Therefore the right hand side of the second rule form must be a sequence of terminal symbols (or a phrase) but not a single symbol (a word). Suppose that the phrase table contains a phrase "am a student" which leads to the following possible tree structure:

Phrase segmentation: "I | am a student"
 Syntactic tree: $(S (NP (NN I)) (VP (VBP am a student)))$
 Rule set: $S \rightarrow NP VP$; $VP \rightarrow VBP$; $NP \rightarrow NN$; $NN \rightarrow I$; $VBP \rightarrow am a student$

We have to find out some way to transform a CFG tree into a tree with phrases at leaves. In the next subsection we propose such an algorithm.

3.2 An Algorithm for Word-to-Phrase Tree Transformation

Table 1 represents our algorithm to transform a CFG tree to a phrase CFG tree. When designing this algorithm, our criterion is to preserve the original structure as much as possible. This algorithm includes two steps. There are a number of notions concerning this algorithm:

- A CFG rule has a head symbol on the right hand side. Using this information, head child of a node on a syntactic tree can be determined.

+ Input:	A CFG tree, a phrase segmentation
+ Output:	A phrase CFG tree
+ Step 1:	Allocate phrases to leaf nodes in a top-down manner: A phrase is allocated to head word of a node if the phrase contains the head word. This head word is then considered as the phrase head.
+ Step 2:	Transform the syntactic tree by replacing leaf nodes by their allocated phrase and removing all nodes whose span is a substring of phrases.

Table 1: An algorithm to transform a CFG tree to a phrase CFG tree.

- If a node is a pre-terminal node (containing POS tag), its head word is itself. If a node is an inner node (containing syntactic constituent tag), its head word is retrieved through the head child.
- Word span of a node is a string of its leaves. For instance, word span of subtree (NP (PRP\$ your) (NN class)) is "your class".

Now we consider an example depicted in Figure 3 and 4. Head children are tagged with functional label H. There are two phrases: "is a" and "in your class". After the Step 1, the phrase "is a" is attached to (VBZ is). The phrase "in your class" is attached to (IN in). In Step 2, the node (V is) is replaced by (V "is a") and (DT a) is removed from its father NP. Similarly, (IN in) is replaced by (IN "in your class") and the subtree NP on the right is removed.

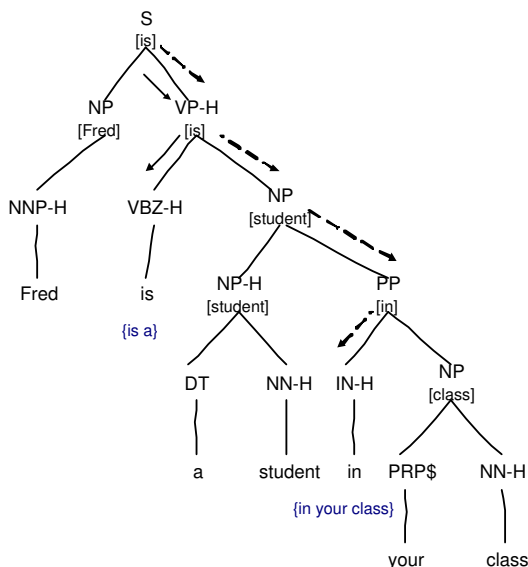


Figure 3: Tree transformation - step 1. Solid arrows show the allocation process of "is a". Dotted arrows demonstrate the allocation process of "in your class"

The proposed algorithm has some properties. We state these properties without presenting proof².

- **Uniqueness:** Given a CFG tree and a phrase segmentation, by applying Algorithm 1, one and only one phrase tree is generated.

²Proofs are simple.

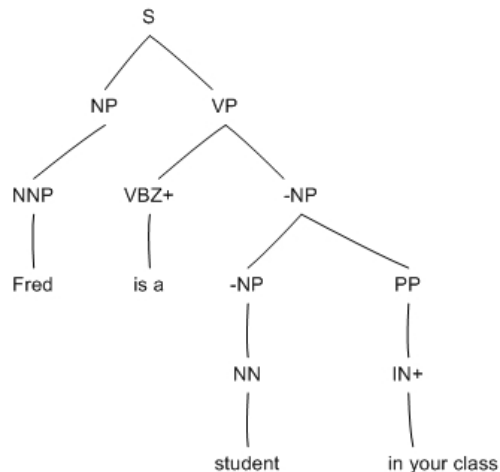


Figure 4: Tree transformation - step 2.

- **Constituent subgraph:** A phrase CFG tree is a connected subgraph of input tree if leaves are ignored.
- **Flatness:** A phrase CFG tree is flatter than input tree.
- **Outside head:** The head of a phrase is always a word whose head outside the phrase. If there is more than one word satisfying this condition, the word at the highest level is chosen.
- **Dependency subgraph:** Dependency graph of a phrase CFG tree is a connected subgraph of input tree's dependency graph if there exist no detached nodes.

The meaning of uniqueness property is that our algorithm is a deterministic procedure. The constituent-subgraph property will be employed in the next section for an efficient decoding algorithm. When a syntactic tree is transformed, a number of subtrees are replaced by phrases. The head word of a phrase is the contact point of that phrase with the remaining part of a sentence. From the dependency point of view, a head word should depend on an outer word rather than an inner word. About dependency-subgraph property, when there is a detached node, an indirect dependency will become a direct one. In any cases, there is no

change in dependency direction. We can observe dependency trees in Figure 5. The first two trees are source dependency tree and phrase dependency tree of the previous example. The last one corresponds to the case in which a detached node exists.

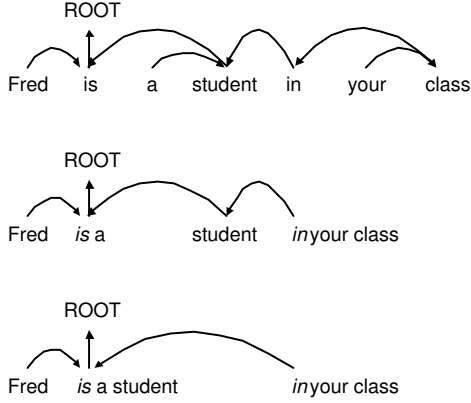


Figure 5: Dependency trees. The third tree corresponds with phrase segmentation: "Fred | is a student | in your class"

3.3 Probabilistic Word-to-Phrase Tree Transformation

We have proposed an algorithm to create a phrase CFG tree from a pair of CFG tree and phrase segmentation. Two questions naturally arise: "is there a way to evaluate how good a phrase tree is?" and "is such an evaluation valuable?" Note that phrase trees are the means to reorder the source sentence represented as phrase segmentations. Therefore a phrase tree is surely not good if no right order can be generated. Now the answer to the second question is clear. We need an evaluation method to prevent our program from generating bad phrase trees. In other words, good phrase trees should be given a higher priority.

We define the phrase tree probability as the product of its rule probability given the original CFG rules:

$$P(T') = \prod_i P(LHS_i \rightarrow RHS'_i | LHS_i \rightarrow RHS_i) \quad (3)$$

where T' is a phrase tree whose CFG rules are $LHS_i \rightarrow RHS'_i$. $LHS_i \rightarrow RHS_i$ are original CFG rules. RHS'_i are subsequences of RHS_i . Since phrase tree rules should capture changes made by the transformation from word to phrase, we use '+' to represent an expansion and '-' to show an overlap. These symbols will be added to a nonterminal on the side having a change. In the previous example, since a head noun in the word tree

has been expanded on the right, the corresponding symbol in phrase tree is NN-H+. A nonterminal X can become one of the following symbols $X, -X, +X, X-, X+, -X-, -X+, +X-, +X+$.

Conditional probabilities are computed in a separate training phase using a source-parsed and word-aligned bitext. First, all phrase pairs consistent with the word alignment are collected. Then using this phrase segmentation and syntactic trees we can generate phrase trees by word-to-phrase tree transformation and extract rules.

4 Phrase Reordering Model

Reordering rules are represented as SCFG rules which can be un-lexicalized or source-side lexicalized (Nguyen and Shimazu, 2006). In this paper, we used un-lexicalized rules. We used a learning algorithm as in (Nguyen and Shimazu, 2006) to learn weighted SCFGs. The training requirements include a bilingual corpus, a word alignment tool, and a broad coverage parser of the source language. The parser is a constituency analyzer which can produce parse tree in Penn Tree-bank's style. The model is applicable to language pairs in which the target language is poor in resources. We used phrase reorder rules whose '+' and '-' symbols are removed.

5 Decoding

A source sentence can have many possible phrase segmentations. Each segmentation in combination with a source tree corresponds to a phrase tree. A phrase-tree forest is a set of those trees. A naive decoding algorithm is that for each segmentation, a phrase tree is generated and then the sentence is translated. This algorithm is very slow or even intractable. Based on the constituent-subgraph property of the tree transformation algorithm, the forest of phrase trees will be packed into a tree-structure container whose backbone is the original CFG tree.

5.1 Translation Options

A translation option encodes a possibility to translate a source phrase (at a leaf node of a phrase tree) to another phrase in target language. Since our decoder uses a log-linear translation model, it can exploit various features of translation options. We use the same features as (Koehn et al., 2003). Basic information of a translation option includes:

- source phrase
- target phrase
- phrase translation score (2)

- lexical translation score (2)
- word penalty

Translation options of an input sentence are collected before any decoding takes place. This allows a faster lookup than consulting the whole phrase translation table during decoding. Note that the entire phrase translation table may be too big to fit into memory.

5.2 Translation Hypotheses

A translation hypothesis represents a partial or full translation of an input sentence. Initial hypotheses correspond to translation options. Each translation hypothesis is associated with a phrase-tree node. In other words, a phrase-tree node has a collection of translation hypotheses. Now we consider basic information contained in a translation hypothesis:

- the cost so far
- list of child hypotheses
- left language model state and right language model state

5.3 Decoding Algorithm

First we consider structure of a syntactic tree. A tree node contains fields such as syntactic category, child list, and head child index. A leaf node has an additional field of word string. In order to extend this structure to store translation hypotheses, a new field of hypothesis collection is appended. A hypothesis collection contains translation hypotheses whose word spans are the same. Actually, it corresponds to a phrase-tree node. A hypothesis collection whose word span is $[i_1, i_2]$ at a node whose tag is X expresses that:

- There is a phrase-tree node (X, i_1, i_2) .
- There exist a phrase $[i_1, i_2]$ or
- There exist a subsequence of X 's child list: $(Y_1, j_0, j_1), (Y_2, j_1 + 1, j_2), \dots, (Y_n, j_{n-1} + 1, j_n)$ where $j_0 = i_1$ and $j_n = i_2$
- Suppose that $[i, j]$ is X 's span, then $[i_1, i_2]$ is a valid phrase node's span if and only if: $i_1 \leq i$ or $i < i_1 \leq j$ and there exist a phrase $[i_0, i_1 - 1]$ overlapping X 's span at $[i, i_1 - 1]$. A similar condition is required of j .

Table 2 shows our decoding algorithm. Step 1 distributes translation options to leaf nodes using a procedure similar to Step 1 of algorithm in Table 1. Step

Corpus	Size	Training	Development	Testing
Conversation	16,809	15,734	403	672
Reuters	57,778	55,757	1,000	1,021

Table 3: Corpora and data sets.

	English	Vietnamese
Sentences		16,809
Average sent. len.	8.5	8.0
Words	143,373	130,043
Vocabulary	9,314	9,557
	English	Japanese
Sentences		57,778
Average sent. len.	26.7	33.5
Words	1,548,572	1,927,952
Vocabulary	31,702	29,406

Table 4: Corpus statistics of translation tasks.

2 helps check valid subsequences in Step 3 fast. Step 3 is a bottom-up procedure, a node is translated if all of its child nodes have been translated. Step 3.1 calls syntactic transformation models. After reordered in Step 3.2, a subsequence will be translated in Step 3.3 using a simple monotonic decoding procedure resulting in new translation hypotheses. We used a beam pruning technique to reduce the memory cost and to accelerate the computation.

6 Experimental Results

6.1 Experimental Settings

We used Reuters³, an English-Japanese bilingual corpus, and Conversation, an English-Vietnamese corpus (Table 4). These corpora were split into data sets as shown in Table 3. Japanese sentences were analyzed by ChaSen⁴, a word-segmentation tool.

A number of tools were used in our experiments. Vietnamese sentences were segmented using a word-segmentation program (Nguyen et al., 2003). For learning phrase translations and decoding, we used Pharaoh (Koehn, 2004), a state-of-the-art phrase-based SMT system which is available for research purpose. For word alignment, we used the GIZA++ tool (Och and Ney, 2000). For learning language models, we used SRILM toolkit (Stolcke, 2002). For MT evaluation, we used BLEU measure (Papineni et al., 2001) calculated by the NIST script version 11b. For the parsing task, we used Charniak's parser (Charniak, 2000). For experiments with chunking (or shallow parsing), we used a CRFs-based chunking tool⁵ to split a source sentence into syntactic chunks. Then a pseudo CFG rule over chunks is built to generate a two-level syntactic tree. This tree can be used in the

³<http://www2.nict.go.jp/x/x161/members/mutiyama/index.html>

⁴<http://chasen.aist-nara.ac.jp/chasen/distribution.html>

⁵<http://crfpp.sourceforge.net/>

+ Input:	A source CFG tree, a translation-option collection
+ Output:	The best target sentence
+ Step 1:	Allocate translation options to hypothesis collections at leaf nodes.
+ Step 2:	Compute overlap vector for all nodes.
+ Step 3:	For each node, if all of its children have been translated, then for each valid sub-sequence of child list, carry out the following steps:
+ Step 3.1:	Retrieve transformation rules
+ Step 3.2:	Reorder the sub-sequence
+ Step 3.3:	Translate the reordered sub-sequence and update corresponding hypothesis collections

Table 2: A bottom-up dynamic-programming decoding algorithm.

Corpus	CFG	PhraseCFG	W2PTT	Reorder
Conversation	2,784	2,684	8,862	2,999
Reuters	7,668	5,479	13,458	7,855

Table 5: Rule induction statistics.

Corpus	Pharaoh	PB system	SD system (chunking)	SD system (full-parsing)
Conversation	35.47	35.66	36.85	37.42
Reuters	24.41	24.20	20.60	25.53

Table 6: BLEU score comparison between phrase-based SMT and syntax-directed SMT. PB=phrase-based; SD=syntax-directed

same way as trees produced by Charniak’s parser.

We built a SMT system for phrase-based log-linear translation models. This system has two decoders: beam search and syntax-based. We implemented the algorithm in Section 5 for the syntax-based decoder. We also implemented a rule induction module and a module for minimum error rate training. We used the system for our experiments reported later.

6.2 Rule Induction

In Table 5, we report statistics of CFG rules, phrase CFG rules, word-to-phrase tree transformation (W2PTT) rules, and reordering rules. All counted rules were in un-lexicalized form. Those numbers are very small in comparison with the number of phrasal translations (up to hundreds of thousands on our corpora). There were a number of “un-seen” CFG rules which did not have a corresponding reordering rule. A reason is that those rules appeared once or several times in the training corpus; however, their hierarchical alignments did not satisfy the conditions for inducing a reordering rule since word alignment is not perfect (Nguyen and Shimazu, 2006). Another reason is that there were CFG rules which required nonlocal reordering. This may be an issue for future research: a Markovization technique for SCFGs.

6.3 BLEU Scores

Table 6 shows a comparison of BLEU scores between Pharaoh, our phrase-based SMT system, and

our syntax-directed (SD) SMT system with chunking and full parsing respectively. On both Conversation corpus and Reuters corpus: The BLEU score of our phrase-based SMT system is comparable to that of Pharaoh; The BLEU score of our SD system with full parsing is higher than that of our phrase-based system. On Conversation corpus, our SD system with chunking has a higher performance in terms of BLEU score than our phrase-based system. Using sign test (Lehmann, 1986), we verified the improvements are statistically significant. However, on Reuters corpus, performance of the SD system with chunking is much lower than the phrase-based system’s. The reason is that in English-Japanese translation, chunk is a too shallow syntactic structure to capture word order information. For example, a prepositional chunk often includes only preposition and adverb, therefore such information does not help reordering prepositional phrases.

6.4 The Effectiveness of the W2PTT Model

Without this feature, BLEU scores decreased around 0.5 on both corpora. We now consider a linguistically motivated example of English-Vietnamese translation to show that phrase segmentation can be evaluated through phrase tree scoring. This example was extracted from Conversation test set.

$$\begin{aligned}
 &\text{English sentence: } \textit{for my wife 's mother} \\
 &\text{Vietnamese word order: } \textit{for mother 's wife my} \\
 &\text{Phrase segmentation 1: } \textit{for my wife | 's | mother} \\
 P1 = &P((PP \rightarrow IN+ -NP | PP \rightarrow IN NP) \times P(-NP \rightarrow NP \textit{NN} | NP \rightarrow NP \\
 &\textit{NN}) \times P(-NP \rightarrow POS | NP \rightarrow PRP\$ \textit{NN} \\
 POS) = &\log(0.00001) + \log(0.14) + \log(0.048) = -5.0.85 - 1.32 = -7.17 \\
 &\text{Phrase segmentation 2: } \textit{for | my wife 's | mother} \\
 P2 = &P((PP \rightarrow IN NP | PP \rightarrow IN NP) \times P(NP \rightarrow NP \textit{NN} | NP \rightarrow NP \\
 &\textit{NN}) \times P(NP \rightarrow POS | NP \rightarrow PRP\$ \textit{NN} POS) \\
 = &\log(0.32) + \log(0.57) + \log(0.048) = -0.5.0.24 - 1.32 = -2.06
 \end{aligned}$$

The first phrase segmentation is bad (or even unacceptable) since the right word order can not be achieved from this segmentation by phrase reordering and word reordering within phrases. The second phrase segmentation is much better. Source syntax tree and phrase trees are shown in Figure 6. The first phrase tree has a much smaller probability (P1=-7.17) than the second (P2=-2.06).

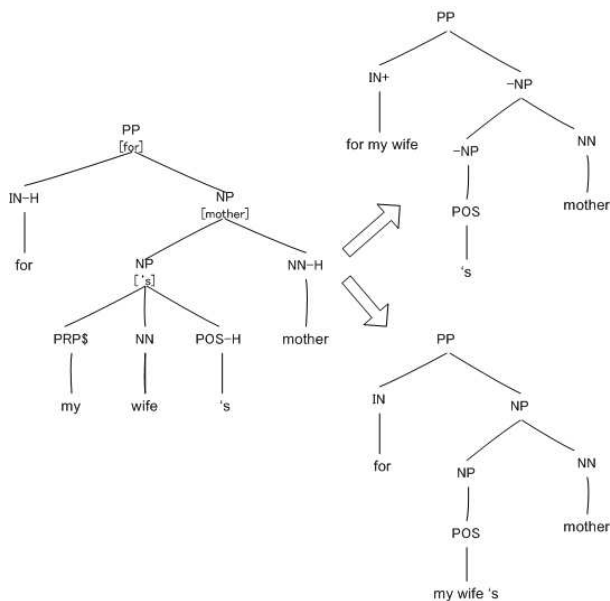


Figure 6: Two phrase trees.

Corpus	Level-1	Level-2	Level-3	Level-4	Full
Conversation	36.85	36.91	37.11	37.23	37.42
Reuters	20.60	22.76	24.49	25.12	25.53

Table 7: BLEU score with different syntactic levels. Level- i means syntactic transformation was applied to tree nodes whose level smaller than or equal to i . The level of a pre-terminal node (POS tag) is 0. The level of an inner node is the maximum of its children's levels.

6.5 Levels of Syntactic Analysis

Since in practice, chunking and full parsing are often used, in Table 6, we showed translation quality of the two cases. It is interesting if we can find how syntactic analysis can affect BLEU score at more intermediate levels (Table 7). On the Conversation corpus, using syntax trees of level-1 is effective in comparison with baseline. The increase of syntactic level makes a steady improvement in translation quality. Note that when we carried out experiments with chunking (considered as level-1 syntax) the translation speed (including chunking) of our tree-to-string system was much faster than baseline systems'. This is an option for developing applications which require high speed such as web translation.

7 Related Works

7.1 A Comparison of Syntactic SMT Methods

To advance the state of the art, SMT system designers have experimented with tree-structured translation models. The underlying computational models

were synchronous context-free grammars and finite-state tree transducers which conceptually have a better expressive power than finite-state transducers. We create Tables 8 and 9 in order to compare syntactic SMT methods including ours. The first row is a baseline phrasal SMT approach. The second column in Table 8 only describes input types because the output is often string. Syntactic SMT methods are different in many aspects. Methods which make use of phrases (in either explicit or implicit way) can beat the baseline approach (Table 8) in terms of BLEU metric. Two main problems these models aim to deal with are word order and word choice. In order to accomplish this purpose, the underlying formal grammars (including synchronous context-free grammars and tree transducers) can be fully lexicalized or unlexicalized (Table 9).

7.2 Non-constituent Phrasal Translations

Liu et al. (2007) proposed forest-to-string rules to capture non-constituent phrasal translation while our system can naturally make use of such kind of phrasal translation by using word-to-phrase tree transformation. Liu et al. (2007) also discussed about how the phenomenon of non-syntactic bilingual phrases is dealt with in other SMT methods. Galley et al. (2006) handled non-constituent phrasal translation by traversing the tree upwards until reaches a node that subsumes the phrase. Marcu et al. (2006) reported that approximately 28% of bilingual phrases are non-syntactic on their English-Chinese corpus. They proposed using a pseudo nonterminal symbol that subsumes the phrase and corresponding multi-headed syntactic structure. One new xRs rule is required to explain how the new nonterminal symbol can be combined with others. This technique brought a significant improvement in performance to their string-to-tree noisy channel SMT system.

8 Conclusions

We have presented a general tree-to-string phrase-based method. This method employs a syntax-based reordering model in the decoding phase. By word-to-phrase tree transformation, all possible phrases are considered in translation. Our method does not suppose a uniform distribution over all possible phrase segmentations as (Koehn et al., 2003) since each phrase tree has a probability. We believe that other kinds of translation unit such as n-gram (Jos et al., 2006), factored phrasal translation (Koehn and Hoang, 2007), or treelet (Quirk et al., 2005) can be used in this method. We would like to consider this problem as a future study. Moreover we would like to use n-best trees as the input of our system. A number

Method	Input	Theoretical model	Decoding style	Linguistic information	Phrase usage	Performance
Koehn et al. (2003)	string	FSTs	beam search	no	yes	baseline
Yamada and Knight (2001)	string	SCFGs	parsing	target	no	not better
Melamed (2003)	string	SCFGs	parsing	both sides	no	not better
Chiang (2005)	string	SCFGs	parsing	no	yes	better
Quirk et al. (2005)	dep. tree	TTs	parsing	source	yes	better
Galley et al. (2006)	string	TTs	parsing	target	yes	better
Liu et al. (2006)	tree	TTs	tree transf.	source	yes	better
Our work	tree	SCFGs	tree transf.	source	yes	better

Table 8: A comparison of syntactic SMT methods (part 1). FST=Finite State Transducer; SCFG=Synchronous Context-Free Grammar; TT=Tree Transducer.

Method	Rule form	Rule function	Rule lexicalization level
Koehn et al. (2003)	no	no	no
Yamada and Knight (2001)	SCFG rule	reorder and function-word ins./del.	unlexicalized
Melamed (2003)	SCFG rule	reorder and word choice	full
Chiang (2005)	SCFG rule	reorder and word choice	full
Quirk et al. (2005)	Treelet pair	word choice	full
Galley et al. (2006)	xRs rule	reorder and word choice	full
Liu et al. (2006)	xRs rule	reorder and word choice	full
Our work	SCFG rule	reorder	unlexicalized

Table 9: A comparison of syntactic SMT methods (part 2). xRs is a kind of rule which maps a syntactic pattern to a string, for example $VP(AUX(does), RB(not), x_0:VB) \rightarrow ne, x_0, pas$. In the column Rule lexicalization level: full=lexicalization using vocabularies of both source language and target language.

of non-local reordering phenomena such as adjunct attachment should be handled in the future.

References

- Charniak, E. 2000. A maximum entropy inspired parser. In *Proceedings of HLT-NAACL*.
- Galley, M., Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeeffe, Wei Wang, Ignacio Thayer 2006. Scalable Inference and Training of Context-Rich Syntactic Translation Models. In *Proceedings of ACL*.
- Jos B. Mario, Rafael E. Banchs, Josep M. Crego, Adri de Gispert, Patrik Lambert, Jos A. R. Fonollosa, Marta R. Costa-juss. 2006. N-gram-based Machine Translation. *Computational Linguistics*, 32(4): 527–549.
- Koehn, P. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of AMTA*.
- Koehn, P. and Hieu Hoang. 2007. Factored Translation Models. In *Proceedings of EMNLP*.
- Koehn, P., F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*.
- Lehmann, E. L. 1986. *Testing Statistical Hypotheses* (Second Edition). Springer-Verlag.
- Liu, Y., Qun Liu, Shouxun Lin. 2006. Tree-to-String Alignment Template for Statistical Machine Translation. In *Proceedings of ACL*.
- Liu, Y., Yun Huang, Qun Liu, and Shouxun Lin 2007. Forest-to-String Statistical Translation Rules. In *Proceedings of ACL*.
- Marcu, D., Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical Machine Translation with Syntactified Target Language Phrases. In *Proceedings of EMNLP*.
- Melamed, I. D. 2004. Statistical machine translation by parsing. In *Proceedings of ACL*.
- Nguyen, Thai Phuong and Akira Shimazu. 2006. Improving Phrase-Based Statistical Machine Translation with Morphosyntactic Transformation. *Machine Translation*, 20(3): 147–166.
- Nguyen, Thai Phuong, Nguyen Van Vinh and Le Anh Cuong. 2003. Vietnamese Word Segmentation Using Hidden Markov Model. In *Proceedings of International Workshop for Computer, Information, and Communication Technologies in Korea and Vietnam*.
- Och, F. J. and H. Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL*.
- Papineni, K., S. Roukos, T. Ward, W.-J. Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. Technical Report RC22176 (W0109-022), IBM Research Report.
- Quirk, C., A. Menezes, and C. Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of ACL*.
- Stolcke, A. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proc. Intl. Conf. Spoken Language Processing*.
- Yamada, K. and K. Knight. 2001. A syntax-based statistical translation model. In *Proceedings of ACL*.

Trainable Speaker-Based Referring Expression Generation

Giuseppe Di Fabbrizio and Amanda J. Stent and Srinivas Bangalore

AT&T Labs - Research, Inc.

180 Park Avenue

Florham Park, NJ 07932, USA

{pino,stent,srini}@research.att.com

Abstract

Previous work in referring expression generation has explored general purpose techniques for attribute selection and surface realization. However, most of this work did not take into account: a) stylistic differences between speakers; or b) trainable surface realization approaches that combine semantic and word order information. In this paper we describe and evaluate several end-to-end referring expression generation algorithms that take into consideration speaker style and use data-driven surface realization techniques.

1 Introduction

Natural language generation (NLG) systems have typically decomposed the problem of generating a linguistic expression from a conceptual specification into three major steps: *content planning*, *text planning* and *surface realization* (Reiter and Dale, 2000). The task in content planning is to select the information that is to be conveyed to maximize communication efficiency. The task in text planning and surface realization is to use the available linguistic resources (words and syntax) to convey the selected information using well-formed linguistic expressions.

During a discourse (whether written or spoken, monolog or dialog), a number of entities are introduced into the discourse context shared by the reader/hearer and the writer/speaker. Constructing linguistic references to these entities efficiently and effectively is a problem that touches on all

parts of an NLG system. Traditionally, this problem is split into two parts. The task of selecting the attributes to use in referring to an entity is the *attribute selection* task, performed during content planning or sentence planning. The actual construction of the referring expression is part of *surface realization*.

There now exist numerous general-purpose algorithms for attribute selection (e.g., (Dale and Reiter, 1995; Krahmer et al., 2003; Belz and Gatt, 2007; Siddharthan and Copestake, 2004)). However, these algorithms by-and-large focus on the algorithmic aspects of referring expression generation rather than on psycholinguistic factors that influence language production. For example, we know that humans exhibit individual differences in language production that can be quite pronounced (e.g. (Belz, 2007)). We also know that the language production process is subject to *lexical priming*, which means that words and concepts that have been used recently are likely to appear again (Levelt, 1989).

In this paper, we look at attribute selection and surface realization for referring expression generation using the TUNA corpus¹, an annotated corpus of human-produced referring expressions that describe furniture and people. We first explore the impact of individual style and priming on attribute selection for referring expression generation. To get an idea of the potential improvement when modeling these factors, we implemented a version of full brevity search that uses speaker-specific constraints, and another version that also uses recency constraints. We found that using speaker-specific constraints led to big performance gains for both TUNA domains, while the use of re-

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

¹<http://www.csd.abdn.ac.uk/research/tuna/>

gency constraints was not as effective for TUNA-style tasks. We then modified Dale and Reiter’s classic attribute selection algorithm (Dale and Reiter, 1995) to model individual differences in style, and found performance gains in this more greedy approach as well.

Then, we look at surface realization for referring expression generation. There are several approaches to surface realizations described in the literature (Reiter and Dale, 2000) ranging from hand-crafted template-based realizers to data-driven syntax-based realizers (Langkilde and Knight, 2000; Bangalore and Rambow, 2000). Template-based realization provides a straightforward method to fill out pre-defined templates with the current attribute values. Data-driven syntax-based methods employ techniques that incorporate the syntactic relations between words which can potentially go beyond local adjacency relations. Syntactic information also helps in eliminating ungrammatical sentence realizations. At the other extreme, there are techniques that exhaustively generate possible realizations with recourse to syntax in as much as it is reflected in local n -grams. Such techniques have the advantage of being robust although they are inadequate to capture long-range dependencies. We explore three techniques for the task of referring expression generation that are different hybrids of hand-crafted and data-driven methods.

The layout of this paper is as follows: In Section 2, we describe the TUNA data set and the task of identifying target entities in the context of distractors. In Section 3, we present our algorithms for attribute selection. Our algorithms for surface realization are presented in Section 4. Our evaluation of these methods for attribute selection and surface realization are presented in Sections 5 and 6.

2 The TUNA Corpus

The TUNA corpus was constructed using a web-based experiment. Participants were presented with a sequence of web pages, on each of which they saw displayed a selection of 7 pictures of either furniture (e.g. Figure 1) or people (e.g. Figure 2) sparsely placed on a 3 row x 5 column grid. One of the pictures (the *target*) was highlighted; the other 6 objects (the *distractors*) were randomly selected from the object database. Participants were told that they were interacting with a

computer system to remove all but the highlighted picture from the screen. They entered a description of the object using natural language to identify the object to the computer system.

The section of the TUNA corpus we used was that provided for the REG 2008 Challenge². The training data includes 319 referring expressions in the furniture domain and 274 in the people domain. The development data (which we used for testing) includes 80 referring expressions in the furniture domain and 68 in the people domain.



Figure 1: Example of data from the furniture domain (*The red couch on top*).



Figure 2: Example of data from the people domain (*The bald subject on the bottom with the white beard*).

3 Attribute Selection Algorithms

Given a set of entities with attributes appropriate to a domain (e.g., cost of flights, author of a book,

²<http://www.nltg.brighton.ac.uk/research/reg08/>. Preliminary versions of these algorithms were used in this challenge and presented at INLG 2008.

color of a car) that are in a discourse context, and a target entity that needs to be identified, the task of attribute selection is to select a subset of the attributes that uniquely identifies the target entity. (Note that there may be more than one such attribute set.) The efficacy of attribute selection can be measured based on the minimality of the selected attribute set as well as its ability to determine the target entity uniquely. There are variations however in terms of what makes an attribute set more preferable to a human. For example, in a people identification task, attributes of faces are generally more memorable than attributes pertaining to outfits. In this paper, we demonstrate that the attribute set is speaker dependent.

In this section, we present two different attribute selection algorithms. The **Full Brevity** algorithm selects the attribute set by exhaustively searching through all possible attribute sets. In contrast, **Dale and Reiter** algorithm orders the attributes based on a heuristic (motivated by human preference) and selects the attributes in that order until the target entity is uniquely determined. We elaborate on these algorithms below.

Full Brevity (FB) We implemented a version of full brevity search. It does the following: first, it constructs \mathcal{AS} , the set of attribute sets that uniquely identify the referent given the distractors. Then, it selects an attribute set $AS_u \in \mathcal{AS}$ based on one of the following four criteria: 1) The **minimality (FB-m)** criterion selects from among the smallest elements of \mathcal{AS} at random. 2) The **frequency (FB-f)** criterion selects the element of \mathcal{AS} that occurred most often in the training data. 3) The **speaker frequency (FB-sf)** criterion selects the element of \mathcal{AS} used most often by this speaker in the training data, backing off to FB-f if necessary. This criterion models individual speaking/writing style. 4) Finally, the **speaker recency (FB-sr)** criterion selects the element of \mathcal{AS} used most recently by this speaker in the training data, backing off to FB-sf if necessary. This criterion models priming.

Dale and Reiter We implemented two variants of the classic Dale & Reiter attribute selection (Dale and Reiter, 1995) algorithm. For **Dale & Reiter basic (DR-b)**, we first build the preferred list of attributes by sorting the attributes according to frequency of use in the training data. We keep separate lists based on the “LOC” condition (if its

value was “+LOC”, the participants were told that they could refer to the target using its location on the screen; if it was “-LOC”, they were instructed not to use location on the screen) and backoff to a global preferred attribute list if necessary. Next, we iterate over the list of preferred attributes and select the next one that rules out at least one entity in the contrast set until no distractors are left. **Dale & Reiter speaker frequency (DR-sf)** uses a different preferred attribute list for each speaker, backing off to the DR-b preferred list if an attribute has never been observed in the current speaker’s preferred attribute list. For the purpose of this task, we did not use any external knowledge (e.g. taxonomies).

4 Surface Realization Approaches

A surface realizer for referring expression generation transforms a set of attribute-value pairs into a linguistically well-formed expression. Our surface realizers, which are all data-driven, involve four stages of processing: (a) lexical choice of words and phrases to realize attribute values; (b) generation of a space of surface realizations (T); (c) ranking the set of realizations using a language model (LM); (d) selecting the best scoring realization. In general, the best ranking realization (T^*) is described by equation 1:

$$T^* = \text{Bestpath}(\text{Rank}(T, LM)) \quad (1)$$

We describe three different methods for creating the search space of surface realizations – *Template-based*, *Dependency-based* and *Permutation-based* methods. Although these techniques share the same method for ranking, they differ in the methods used for generating the space of possible surface realizations.

4.1 Generating possible surface realizations

In order to transform the set of attribute-value pairs into a linguistically well-formed expression, the appropriate words that realize each attribute value need to be selected (lexical choice) and the selected words need to be ordered according to the syntax of the target language (lexical order). We present different models for approximating the syntax of the target language. All three models tightly integrate the lexical choice and lexical re-ordering steps.

4.1.1 Template-Based Realizer

In the template-based approach, surface realizations from our training data are used to infer a set of templates. In the TUNA data, each attribute in each referring expression is annotated with its attribute type (e.g. in “the large red sofa” the second word is labeled ‘size’, the third ‘color’ and the fourth ‘type’). We extract the annotated referring expressions from each trial in the training data and replace each attribute value with its type (e.g. “the size color type”) to create a template. Each template is indexed by the lexicographically sorted list of attribute types it contains (e.g. color size type). If an attribute set is not found in the training data (e.g. color size) but a superset of that set is (e.g. color size type), then the corresponding template(s) may be used, with the un-filled attribute types deleted prior to output.

At generation time, we find all possible realizations (l) (from the training data) of each attribute value (a) in the input attribute set (AS), and fill in each possible template (t) with each combination of the attribute realizations. The space of possible surface realizations is represented as a weighted finite-state automaton. The weights are computed from the prior probability of each template and the prior probability of each lexical item realizing an attribute (Equation 2). We have two versions of this realizer: one with speaker-specific lexicons and templates (**Template-S**), and one without (**Template**). We report results for both.

$$P(T|AS) = \sum_t P(t|AS) * \prod_{a \in t} \sum_l P(l|a, t) \quad (2)$$

4.1.2 Dependency-Based Realizer

To construct our dependency-based realizer, we first parse all the word strings from the training data using the dependency parser described in (Bangalore et al., 2005; Nasr and Rambow, 2004). Then, for every pair of words w_i, w_j that occur in the same referring expression (RE) in the training data, we compute: $freq(i < j)$, the frequency with which w_i precedes w_j in any RE; $freq(dep(w_i, w_j) \wedge i < j)$, the frequency with which w_i depends on and precedes w_j in any RE, and $freq(dep(w_i, w_j) \wedge j < i)$, the frequency with which w_i depends on and follows w_j in any RE.

At generation time, we find all possible realizations of each attribute value in the input attribute

set, and for each combination of attribute realizations, we find the most likely set of dependencies and precedences given the training data. In other words, we bin the selected attribute realizations according to whether they are most likely to precede, depend on and precede, depend on and follow, or follow, the head word they are closest to. The result is a set of weighted partial orderings on the attribute realizations. As with the template-based surface realizer, we implemented speaker-specific and speaker-independent versions of the dependency-based surface realizer. Once again, we encode the space of possible surface realizations as a weighted finite-state automaton.

4.1.3 Permute and Rank Realizer

In this method, the lexical items associated with each attribute value to be realized are treated as a disjunctive set of tokens. This disjunctive set is represented as a finite-state automaton with two states and transitions between them labeled with the tokens of the set. The transitions are weighted by the negative logarithm of the probability of the lexical token (l) being associated with that attribute value (a): $(-\log(P(l|a)))$. These sets are treated as bags of tokens; we create permutations of these bags of tokens to represent the set of possible surface realizations.

In general, the number of states of the minimal permutation automaton of even a linear automaton (finite-state representation of a string) grows exponentially with the number of words of the string. Although creating the full permutation automaton for full natural language generation tasks could be computationally prohibitive, most attribute sets in our two domains contain no more than five attributes. So we choose to explore the full permutation space. A more general approach might constrain permutations to be within a local window of adjustable size (also see (Kanthak et al., 2005)).

Figure 3 shows the minimal permutation automaton for an input sequence of 4 words and a window size of 2. Each state of the automaton is indexed by a bit vector of size equal to the number of words/phrases of the target sentence. Each bit of the bit vector is set to 1 if the word/phrase in that bit position is used on any path from the initial to the current state. The next word for permutation from a given state is restricted to be within the window size (2 in our case) positions counting from the first as-yet uncovered position in that state. For example, the state indexed with vector “1000” rep-

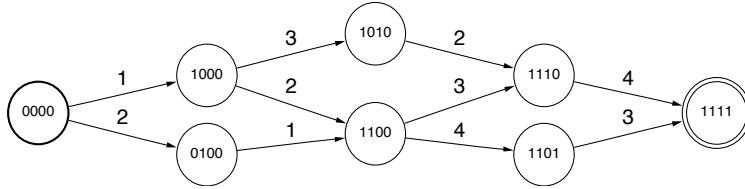


Figure 3: Locally constraint permutation automaton for a sentence with 4 positions and a window size of 2.

resents the fact that the word/phrase at position 1 has been used. The next two (window=2) positions are the possible outgoing arcs from this state with labels 2 and 3 connecting to state “1100” and “1010” respectively. The bit vectors of two states connected by an arc differ only by a single bit. Note that bit vectors elegantly solve the problem of recombining paths in the automaton as states with the same bit vectors can be merged. As a result, a fully minimized permutation automaton has only a single initial and final state.

4.2 Ranking and Recovering a Surface Realization

These three methods for surface realization create a space of possible linguistic expressions given the set of attributes to be realized. These expressions are encoded as finite-state automata and have to be ranked based on their syntactic well-formedness. We approximate the syntactic well-formedness of an expression by the n -gram likelihood score of that expression. We use a trigram model trained on the realizations in the training corpus. This language model is also represented as a weighted finite-state automaton. The automaton representing the space of possible realizations and the one representing the language model are composed. The result is an automaton that ranks the possible realizations according to their n -gram likelihood scores. We then produce the best-scoring realization as the target realization of the input attribute set.

We introduce a parameter λ which allows us to control the importance of the prior score relative to the language model scores. We weight the finite-state automata according to this parameter as shown in Equation 3.

$$T^* = \text{Bestpath}(\lambda * T \circ (1 - \lambda) * LM) \quad (3)$$

	DICE	MASI	Acc.	Uniq.	Min.
Furniture					
FB-m	.36	.16	0	1	1
FB-f	.81	.58	.40	1	0
FB-sf	.95	.87	.79	1	0
FB-sr	.93	.81	.71	1	0
DR-b	.81	.60	.45	1	0
DR-sf	.86	.64	.45	1	.04
People					
FB-m	.26	.12	0	1	1
FB-f	.58	.37	.28	1	0
FB-sf	.94	.88	.84	1	.01
FB-sr	.93	.85	.79	1	.01
DR-b	.70	.45	.25	1	0
DR-sf	.78	.55	.35	1	0
Overall					
FB-m	.32	.14	0	1	1
FB-f	.70	.48	.34	1	0
FB-sf	.95	.87	.81	1	.01
FB-sr	.93	.83	.75	1	.01
DR-b	.76	.53	.36	1	0
DR-sf	.82	.60	.41	1	.02

Table 1: Results for attribute selection

5 Attribute Selection Experiments

Data Preparation The training data were used to build the models outlined above. The development data were then processed one-by-one.

Metrics We report performance using the metrics used for the REG 2008 competition. The MASI metric is a metric used in summarization that measures agreement between two annotators (or one annotator and one system) on set-valued items (Nenkova et al., 2007). Values range from 0 to 1, with 1 representing perfect agreement. The DICE metric is also a measure of association whose value varies from 0 (no association) to 1 (total association) (Dice, 1945). The Accuracy metric is binary-valued: 1 if the attribute set is identical to that selected by the human, 0 otherwise. The Uniqueness metric is also binary-valued: 1 if the attribute set uniquely identifies the target referent among the distractors, 0 otherwise. Finally, the Minimality metric is 1 if the selected attribute set is as small as possible (while still uniquely identifying the target referent), and 0 otherwise. We note

that attribute selection algorithms such as Dale & Reiter’s are based on the observation that humans frequently do not produce minimal referring expressions.

Results Table 1 shows the results for variations of full brevity. As we would expect, all approaches achieve a perfect score on uniqueness. For both corpora, we see a large performance jump when we use speaker constraints for all metrics other than minimality. However, when we incorporate recency constraints as well performance declines slightly. We think this is due to two factors: first, the speakers are not in a conversation, and self-priming may have less impact than other-priming; and second, we do not always have the most recent prior utterance for a given speaker in the training data.

Table 1 also shows the results for variations of Dale & Reiter’s algorithm. When we incorporate speaker constraints, we again see a performance jump for most metrics, although compared to the best possible case (full brevity) there is still room for improvement.

We conclude that speaker constraints can be successfully used in standard attribute selection algorithms to improve performance on this task.

The most relevant previous research is the work of (Gupta and Stent, 2005), who modified Dale and Reiter’s algorithm to model speaker adaptation in dialog. However, this corpus does not involve dialog so there are no cross-speaker constraints, only within-speaker constraints (speaker style and priming).

6 Surface Realization Experiments

Data Preparation We first normalized the training data to correct misspellings and remove punctuation and capitalization. We then extracted a phrasal lexicon. For each attribute value we extracted the count of all realizations of that value in the training data. We treated locations as a special case, storing separately the realizations of x-y coordinate pairs and single x- or y-coordinates. We added a small number of realizations by hand to cover possible attribute values not seen in the training data.

Realization We ran two realization experiments. In the first experiment, we used the human-selected attribute sets in the development data as the input to realization. If we want to maxi-

	λ	SED	ACC	Bleu	NIST
Furniture					
Permute&Rank	0.01	3.54	0.14	0.311	3.87
Dependency	0.90	4.51	0.09	0.206	3.29
Dependency-S	0.60	4.30	0.11	0.232	3.91
Template	0.10	3.59	0.13	0.328	3.93
Template-S	0.10	2.80	0.28	0.403	4.67
People					
Permute&Rank	0.04	4.37	0.10	0.227	3.15
Dependency	0.70	6.10	0.00	0.072	2.35
Dependency-S	0.50	5.84	0.02	0.136	3.05
Template	0.80	3.87	0.07	0.250	3.18
Template-S	0.70	3.79	0.15	0.265	3.59
Overall					
Permute&Rank	.01/0.04	3.92	0.12	0.271	4.02
Dependency	0.9/0.7	5.24	0.05	0.146	3.23
Dependency-S	0.6/0.5	5.01	0.07	0.187	3.98
Template	0.1/0.8	3.77	0.10	0.285	4.09
Template-S	0.1/0.7	3.26	0.22	0.335	4.77

Table 2: Results for realization using speakers’ attribute selection (SED: String Edit Distance, ACC: String Accuracy)

mize humanlikeness, then using these attribute sets should give us an idea of the best possible performance of our realization methods. In the second experiment, we used the attribute sets output by our best-performing attribute selection algorithms (FB-sf and DR-sf) as the input to realization.

Metrics We report performance of our surface realizers using the metrics used for the REG 2008 shared challenge and standard metrics used in the natural language generation and machine translation communities. String Edit Distance (SED) is a measure of the number of words that would have to be added, deleted, or replaced in order to transform the generated referring expression into the one produced by the human. As used in the REG 2008 shared challenge, it is unnormalized, so its values range from zero up. Accuracy (ACC) is binary-valued: 1 if the generated referring expression is identical to that produced by the human (after spelling correction and normalization), and 0 otherwise. Bleu is an n-gram based metric that counts the number of 1, 2 and 3 grams shared between the generated string and one or more (preferably more) reference strings (Papineni et al., 2001). Bleu values are normalized and range from 0 (no match) to 1 (perfect match). Finally, the NIST metric is a variation on the Bleu metric that, among other things, weights rare n-grams higher than frequently-occurring ones (Dodington, 2002). NIST values are unnormalized.

	SED		ACC		Bleu		NIST	
	Furniture							
	FB-sf	DR-sf	FB-sf	DR-sf	FB-sf	DR-sf	FB-sf	DR-sf
Permute&Rank	3.97	4.22	0.09	0.06	.291	.242	3.82	3.32
Dependency	4.80	5.03	0.04	0.03	.193	.105	3.32	2.46
Dependency-S	4.71	4.88	0.06	0.04	.201	.157	3.74	3.26
Template	3.89	4.56	0.09	0.05	.283	.213	3.48	3.22
Template-S	3.26	3.90	0.19	0.12	.362	.294	4.41	4.07
	People							
Permute&Rank	4.75	5.82	0.09	0.03	.171	.110	2.70	2.31
Dependency	6.35	6.91	0.00	0.00	.068	.073	1.81	1.86
Dependency-S	5.94	6.18	0.01	0.00	.108	.113	2.73	2.41
Template	3.62	4.24	0.07	0.04	.231	.138	2.88	1.35
Template-S	3.76	4.38	0.12	0.06	.201	.153	2.76	1.88
	Overall							
Permute&Rank	4.33	4.96	0.09	0.05	.236	.235	3.73	3.72
Dependency	5.51	6.00	0.02	0.01	.136	.091	2.97	2.50
Dependency-S	5.36	5.67	0.04	0.02	.159	.136	3.77	3.25
Template	3.76	4.41	0.08	0.05	.258	.180	3.69	2.89
Template-S	3.48	4.12	0.16	0.09	.288	.229	4.15	3.58

Table 3: Results for realization with different attribute selection algorithms

	Furniture		People	
	FB-sf	DR-sf	FB-sf	DR-sf
Permute&Rank	.01	.05	.05	.04
Dependency	.9	.9	.9	.1
Dependency-S	.2	.2	.4	.4
Template	.8	.8	.8	.8
Template-S	.6	.8	.8	.8

Table 4: Optimal λ values with different attribute selection algorithms

Results Our experimental results are shown in Tables 2 and 3. (These results are the results obtained with the language model weighting that gives best performance; the weights are shown in Tables 2 and 4.) Our approaches work better for the furniture domain, where there are fewer attributes, than for the people domain. For both domains, for automatic and human attribute selection, the speaker-dependent Template-based approach seems to perform the best, then the speaker-independent Template-based approach, and then the Permute&Rank approach. However, we find automatic metrics for evaluating generation quality to be unreliable. We looked at the output of the surface realizers for the two examples in Section 2. The best output for the example in Figure 1 is from the FB-sf template-based speaker-dependent algorithm, which is *the big red sofa*. The worst output is from the DR-sf dependency-based speaker-dependent algorithm, which is *on the left red chair with three seats*. The best output for the example in Figure 2 is from the FB-sf template-based speaker-independent algorithm, which is *the man with the white beard*. The worst output is from the

FB-sf dependency-based speaker-dependent algorithm, which is *beard man white*.

Discussion The Template-S approach achieves the best string edit distance scores, but it is not very robust. If no examples are found in the training data that realize (a superset of) the input attribute set, neither Template approach will produce any output.

The biggest cause of errors for the Permute and Reorder approach is missing determiners and missing modifiers. The biggest cause of errors for the Dependency approach is missing determiners and reordered words. The Template approach sometimes has repeated words (e.g. “middle”, where “middle” referred to both x- and y-coordinates).

Here we report performance using automatic metrics, but we find these metrics to be unreliable (particularly in the absence of multiple reference texts). Also, we are not sure that people would accept from a computer system output that is very human-like in this domain, as the human-like output is often ungrammatical and telegraphic (e.g. “grey frontal table”). We plan to do a human evaluation soon to better analyze our systems’ performance.

7 Conclusions

When building computational models of language, knowledge about the factors that influence human language production can prove very helpful. This knowledge can be incorporated in frequentist and heuristic approaches as constraints or features. In the experiments described in this paper, we used

data-driven, speaker-aware approaches to attribute selection and referring expression realization. We showed that individual speaking style can be usefully modeled even for quite ‘small’ generation tasks, and confirmed that data-driven approaches to surface realization can work well using a range of lexical, syntactic and semantic information.

We plan to explore the impact of human visual search strategies (Rayner, 1998) on the referring expression generation task. In addition, we are planning a human evaluation of the generation systems’ output. Finally, we plan to apply our algorithms to a conversational task.

Acknowledgments

We thank Anja Belz, Albert Gatt, and Eric Kow for organizing the REG competition and providing data, and Gregory Zelinsky for discussions about visually-based constraints.

References

- Bangalore, S. and O. Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proc. COLING*.
- Bangalore, S., A. Emami, and P. Haffner. 2005. Factoring global inference by enriching local representations. Technical report, AT&T Labs-Research.
- Belz, A. and A. Gatt. 2007. The attribute selection for GRE challenge: Overview and evaluation results. In *Proc. UCNLG+MT at MT Summit XI*.
- Belz, A. 2007. Probabilistic generation of weather forecast texts. In *Proc. NAACL/HLT*.
- Dale, R. and E. Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2).
- Dice, L. 1945. Measures of the amount of ecologic association between species. *Ecology*, 26.
- Doddington, G. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. HLT*.
- Gupta, S. and A. Stent. 2005. Automatic evaluation of referring expression generation using corpora. In *Proc. UCNLG*.
- Kanthak, S., D. Vilar, E. Matusov, R. Zens, and H. Ney. 2005. Novel reordering approaches in phrase-based statistical machine translation. In *Proc. ACL Workshop on Building and Using Parallel Texts*.
- Krahmer, E., S. van Erk, and A. Verleg. 2003. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1).
- Langkilde, I. and K. Knight. 2000. Forest-based statistical sentence generation. In *Proc. NAACL*.
- Levelt, W., 1989. *Speaking: From intention to articulation*, pages 222–226. MIT Press.
- Nasr, A. and O. Rambow. 2004. Supertagging and full parsing. In *Proc. 7th International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7)*.
- Nenkova, A., R. Passonneau, and K. McKeown. 2007. The Pyramid method: incorporating human content selection variation in summarization evaluation. *ACM Transactions on speech and language processing*, 4(2).
- Papenini, K., S. Roukos, T. Ward, and W.-J. Zhu. 2001. BLEU: A method for automatic evaluation of machine translation. In *Proc. ACL*.
- Rayner, K. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3).
- Reiter, E. and R. Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.
- Siddharthan, A. and A. Copestake. 2004. Generating referring expressions in open domains. In *Proc. ACL*.

The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies

Mihai Surdeanu^{†,*} Richard Johansson[‡] Adam Meyers[◇]
Lluís Màrquez^{††} Joakim Nivre^{‡‡,**}

[†]: Barcelona Media Innovation Center, mihai.surdeanu@barcelonamedia.org

^{*}: Yahoo! Research Barcelona, mihais@yahoo-inc.com

[‡]: Lund University, richard@cs.lth.se

[◇]: New York University, meyers@cs.nyu.edu

^{††}: Technical University of Catalonia, lluism@lsi.upc.edu

^{‡‡}: Växjö University, joakim.nivre@vxu.se

^{**}: Uppsala University, joakim.nivre@lingfil.uu.se

Abstract

The Conference on Computational Natural Language Learning is accompanied every year by a shared task whose purpose is to promote natural language processing applications and evaluate them in a standard setting. In 2008 the shared task was dedicated to the joint parsing of syntactic and semantic dependencies. This shared task not only unifies the shared tasks of the previous four years under a unique dependency-based formalism, but also extends them significantly: this year's syntactic dependencies include more information such as named-entity boundaries; the semantic dependencies model roles of both verbal and nominal predicates. In this paper, we define the shared task and describe how the data sets were created. Furthermore, we report and analyze the results and describe the approaches of the participating systems.

1 Introduction

In 2004 and 2005 the shared tasks of the Conference on Computational Natural Language Learning (CoNLL) were dedicated to semantic role labeling (SRL), in a monolingual setting (English). In 2006 and 2007 the shared tasks were devoted to the parsing of syntactic dependencies, using corpora from up to 13 languages. The CoNLL-2008 shared task¹ proposes a unified dependency-based

formalism, which models both syntactic dependencies and semantic roles. Using this formalism, this shared task merges both the task of syntactic dependency parsing and the task of identifying semantic arguments and labeling them with semantic roles. Conceptually, the 2008 shared task can be divided into three subtasks: (i) parsing of syntactic dependencies, (ii) identification and disambiguation of semantic predicates, and (iii) identification of arguments and assignment of semantic roles for each predicate. Several objectives were addressed in this shared task:

- SRL is performed and evaluated using a dependency-based representation for both syntactic and semantic dependencies. While SRL on top of a dependency treebank has been addressed before (Hacioglu, 2004), our approach has several novelties: (i) our constituent-to-dependency conversion strategy transforms all annotated semantic arguments in PropBank and NomBank not just a subset; (ii) we address propositions centered around both verbal (PropBank) and nominal (NomBank) predicates.
- Based on the observation that a richer set of syntactic dependencies improves semantic processing (Johansson and Nugues, 2007), the syntactic dependencies modeled are more complex than the ones used in the previous CoNLL shared tasks. For example, we now include apposition links, dependencies derived from named entity (NE) structures, and better modeling of long-distance grammatical relations.
- A practical framework is provided for the joint learning of syntactic and semantic dependencies.

©2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

¹<http://www.yr-bcn.es/conll2008>

Given the complexity of this shared task, we limited the evaluation to a monolingual, English-only setting. The evaluation is separated into two different challenges: a closed challenge, where systems have to be trained strictly with information contained in the given training corpus, and an open challenge, where systems can be developed making use of any kind of external tools and resources. The participants could submit results in either one or both challenges.

This paper is organized as follows. Section 2 defines the task, including the format of the data, the evaluation metrics, and the two challenges. Section 3 introduces the corpora used and our constituent-to-dependency conversion procedure. Section 4 summarizes the results of the submitted systems. Section 5 discusses the approaches implemented by participants. Section 6 analyzes the results using additional non-official evaluation measures. Section 7 concludes the paper.

2 Task Definition

In this section we provide the definition of the shared task, starting with the format of the shared task data, followed by a description of the evaluation metrics used and a discussion of the two shared task challenges, i.e., closed and open.

2.1 Data Format

The data format used in this shared task was highly influenced by the formats used in the 2004–2007 shared tasks. The data follows these general rules:

- The files contain sentences separated by a blank line.
- A sentence consists of one or more tokens and the information for each token is represented on a separate line.
- A token consists of at least 11 fields. The fields are separated by one or more whitespace characters (spaces or tabs). Whitespace characters are not allowed within fields.

Table 1 describes the fields stored for each token in the closed-track data sets. Columns 1–3 and 5–8 are available at both training and test time. Column 4, which contains gold-standard part-of-speech (POS) tags, is not given at test time. The same holds for columns 9 and above, which contain the syntactic and semantic dependency structures that the systems should predict.

The PPOS and PPOSS fields were automatically predicted using the SVMTool POS tagger (Giménez, 2004). To predict the tags in the training set, a 5-fold cross-validation procedure was used. The LEMMA and SPLIT_LEMMA fields were predicted using the built-in lemmatizer in WordNet (Fellbaum, 1998) based on the most frequent sense for the form and part-of-speech tag.

Since NomBank uses a sub-word analysis in some hyphenated words (such as [*finger*]_{ARG}-[*pointing*]_{PRED}), the data format represents the parts in hyphenated words as separate tokens (columns 6–8). However, the format also represents how the parts originally fit together before splitting (columns 2–5). Padding characters (“_”) are used in columns 2–5 to ensure the same number of rows for all columns corresponding to one sentence. All syntactic and semantic dependencies are annotated relative to the split word forms (columns 6–8).

Table 2 shows the columns available to the systems participating in the open challenge: named-entity labels as in the CoNLL-2003 Shared Task (Tjong Kim San and De Meulder, 2003) and from the BBN Wall Street Journal Entity Corpus,² WordNet supersense tags, and the output of an off-the-shelf dependency parser (Nivre et al., 2007b). Columns 1–3 were predicted using the tagger of Ciaramita and Altun (2006). Because the BBN corpus shares lexical content with the Penn Treebank, we generated the BBN tags using a 2-fold cross-validation procedure.

2.2 Evaluation Measures

We separate the evaluation measures into two groups: (i) official measures, which were used for the ranking of participating systems, and (ii) additional unofficial measures, which provide further insight into the performance of the participating systems.

2.2.1 Official Evaluation Measures

The official evaluation measures consist of three different scores: (i) syntactic dependencies are scored using the labeled attachment score (LAS), (ii) semantic dependencies are evaluated using a labeled F₁ score, and (iii) the overall task is scored with a macro average of the two previous scores. We describe all these scoring measures next.

The LAS score is defined similarly as in the previous two shared tasks, as the percentage of to-

²LDC catalog number LDC2005T33.

Number	Name	Description
1	ID	Token counter, starting at 1 for each new sentence.
2	FORM	Unsplit word form or punctuation symbol.
3	LEMMA	Predicted lemma of FORM.
4	GPOS	Gold part-of-speech tag from the Treebank (empty at test time).
5	PPOS	Predicted POS tag.
6	SPLIT_FORM	Tokens split at hyphens and slashes.
7	SPLIT_LEMMA	Predicted lemma of SPLIT_FORM.
8	PPOSS	Predicted POS tags of the split forms.
9	HEAD	Syntactic head of the current token, which is either a value of ID or zero (0).
10	DEPREL	Syntactic dependency relation to the HEAD.
11	PRED	Rolesets of the semantic predicates in this sentence.
12...	ARG	Columns with argument labels for each semantic predicate following textual order.

Table 1: Column format in the closed-track data. The columns in the lower part of the table are unseen at test time and are to be predicted by systems.

Number	Name	Description
1	CONLL2003	Named entity labels using the tag set from the CoNLL-2003 shared task.
2	BBN	NE labels using the tag set from the BBN Wall Street Journal Entity Corpus.
3	WNSS	WordNet super senses.
4	MALT_HEAD	Head of the syntactic dependencies generated by MaltParser.
5	MALT_DEPREL	Label of syntactic dependencies generated by MaltParser.

Table 2: Column format in the open-track data.

kens for which a system has predicted the correct HEAD and DEPREL columns (see Table 1). Same as before, our scorer also computes the unlabeled attachment score (UAS), i.e., the percentage of tokens with correct HEAD, and label accuracy, i.e., the percentage of tokens with correct DEPREL.

The semantic propositions are evaluated by converting them to semantic dependencies, i.e., we create a semantic dependency from every predicate to all its individual arguments. These dependencies are labeled with the labels of the corresponding arguments. Additionally, we create a semantic dependency from each predicate to a virtual ROOT node. The latter dependencies are labeled with the predicate senses. This approach guarantees that the semantic dependency structure conceptually forms a single-rooted, connected (but not necessarily acyclic) graph. More importantly, this scoring strategy implies that if a system assigns the incorrect predicate sense, it still receives some points for the arguments correctly assigned. For example, for the correct proposition:

verb.01: ARG0, ARG1, ARGM-TMP

the system that generates the following output for the same argument tokens:

verb.02: ARG0, ARG1, ARGM-LOC

receives a labeled precision score of 2/4 because two out of four semantic dependencies are incorrect: the dependency to ROOT is labeled 02 in-

stead of 01 and the dependency to the ARGM-TMP is incorrectly labeled ARGM-LOC. Using this strategy we compute precision, recall, and F_1 scores for both labeled and unlabeled semantic dependencies.

Finally, we combine the syntactic and semantic measures into one global measure using macro averaging. We compute macro precision and recall scores by averaging the labeled precision and recall for semantic dependencies with the LAS for syntactic dependencies:³

$$LMP = W_{sem} * LP_{sem} + (1 - W_{sem}) * LAS \quad (1)$$

$$LMR = W_{sem} * LR_{sem} + (1 - W_{sem}) * LAS \quad (2)$$

where LMP is the labeled macro precision and LP_{sem} is the labeled precision for semantic dependencies. Similarly, LMR is the labeled macro recall and LR_{sem} is the labeled recall for semantic dependencies. W_{sem} is the weight assigned to the semantic task.⁴ The macro labeled F_1 score, which was used for the ranking of the participating systems, is computed as the harmonic mean of LMP and LMR .

³We can do this because the LAS for syntactic dependencies is a special case of precision and recall, where the predicted number of dependencies is equal to the number of gold dependencies.

⁴We assign equal weight to the two tasks, i.e., $W_{sem} = 0.5$.

2.2.2 Additional Evaluation Measures

We used several additional evaluation measures to further analyze the performance of the participating systems.

The first additional measure used is *Exact Match*, which reports the percentage of sentences that are completely correct, i.e., all the generated syntactic dependencies are correct and all the semantic propositions are present and correct. While this score is significantly lower than any of the official scores, it will award systems that performed joint learning or optimization for all subtasks.

In the same spirit but focusing on the semantic subtasks, we report the *Perfect Proposition F₁* score, where we score entire semantic frames or propositions. This measure is similar to the PProps accuracy score from the 2005 shared task (Carreras and Màrquez, 2005), with the caveat that this year this score is implemented as an F₁ measure, because predicates are not provided in the test data. Hence, propositions may be over or under generated at prediction time.

Lastly, we analyze systems based on the ratio between labeled F₁ score for semantic dependencies and the LAS for syntactic dependencies. In other words, this measure normalizes the semantic scores relative to the performance of the parsing component. This measure estimates the true overall performance of the semantic subtasks, independent of the syntactic parser.⁵ For example, this score addresses the situations where the semantic labeled F₁ score of one system is artificially low because the corresponding syntactic component does not perform well.

2.3 Closed and Open Challenges

Similarly to the CoNLL-2005 shared task, this shared task evaluation is separated into two challenges:

Closed Challenge - systems have to be built strictly with information contained in the given training corpus, and tuned with the development section. In addition, the PropBank and NomBank lexical frames can be used. These restrictions mean that constituent-based parsers or SRL systems can not be used in this challenge because the constituent-based annotations are not provided in our training set. The aim of this challenge is to

⁵A correct evaluation of the stand-alone SRL systems would require the usage of gold syntactic dependencies, but these were not provided for the testing corpora.

compare the performance of the participating systems in a fair environment.

Open Challenge - systems can be developed making use of any kind of external tools and resources. The only condition is that such tools or resources must not have been developed with the annotations of the test set, both for the input and output annotations of the data. In this challenge, we are interested in learning methods which make use of any tools or resources that might improve the performance. For example, we encourage the use of semantic information, as provided by NE recognition or word-sense disambiguation (WSD) systems (such state-of-the-art annotations are provided by the organizers, see Table 2). Also, in this challenge participants are encouraged to use constituent-based parsers and SRL systems, as long as these systems were trained only with the sections of Penn Treebank used in the shared task training corpus. To encourage the participation of the groups that are only interested in SRL, the organizers provide also the output of a state-of-the-art dependency parser as input in this challenge. The comparison of different systems in this setting may not be fair, and thus ranking of systems is not necessarily important.

3 Data

The corpora used in the shared task evaluation were generated through a process that merges several input corpora and converts them from the constituent-based formalism to dependencies. This section starts with an introduction of the input corpora used, followed by a description of the constituent-to-dependency conversion process. The section concludes with an overview of the shared task corpora.

3.1 Input Corpora

Input to our merging procedures includes the Penn Treebank, BBN's named entity corpus, PropBank and NomBank. In this section, we will provide brief descriptions of these annotations in terms of both form and content. All annotations are currently being distributed by the Linguistic Data Consortium, with the exception of NomBank, which is freely downloadable.⁶

⁶<http://nlp.cs.nyu.edu/meyers/NomBank.html>

3.1.1 Penn Treebank 3

The Penn Treebank 3 corpus (Marcus et al., 1993) consists of hand-coded parses of the Wall Street Journal (test, development and training) and a small subset of the Brown corpus (W. N. Francis and H. Kučera, 1964) (test only). These hand parses are notated in-line and sometimes involve changing the strings of the input data. For example, in file *wsj_0309*, the token *fearlast* in the text corresponds to the two tokens *fear* and *last* in the annotated data. In a similar way, *cannot* is regularly split to *can* and *not*. It is significant that the other annotations assume the tokenization of the Penn Treebank, as this makes it easier for us to merge the annotation. The Penn Treebank syntactic annotation includes phrases, parts of speech, empty category representations of various filler/gap constructions and other phenomena, based on a theoretical perspective similar to that of Government and Binding Theory (Chomsky, 1981).

3.1.2 BBN Pronoun Coreference and Entity Type Corpus

BBN's NE annotation of the Wall Street Journal corpus (Weischedel and Brunstein, 2005) takes the form of SGML inline markup of text, tokenized to be completely compatible with the Penn Treebank annotation, e.g., *fearlast* and *cannot* are split in the same ways. Named entity categories include: Person, Organization, Location, GPE, Facility, Money, Percent, Time and Date, based on the definitions of these categories in MUC (Chinchor and Robinson, 1998) and ACE⁷ tasks. Subcategories are included as well. Note however that from this corpus we only use NE boundaries to derive NAME dependencies between NE tokens, e.g., we create a NAME dependency from *Mary* to *Smith* given the NE mention *Mary Smith*.

3.1.3 Proposition Bank I (PropBank)

The PropBank annotation (Palmer et al., 2005) classifies the arguments of all the main verbs in the Penn Treebank corpus, other than *be*. Arguments are numbered (ARG0, ARG1, ...) based on lexical entries or frame files. Different sets of arguments are assumed for different rolesets. Dependent constituents that fall into categories independent of the lexical entries are classified as various types

of ARGM (TMP, ADV, etc.).⁸ Rather than using PropBank directly, we used the version created for the CoNLL-2005 shared task (Carreras and Màrquez, 2005). PropBank's pointers to subtrees are converted into the list of leaves of those subtrees, minus the empty categories. On occasion, arguments of verbs end up being two non-adjacent substrings. For example, the argument of *claims* in the following sentence is indicated in bold: **This sentence, Mary claims, is self-referential.** The CoNLL-2005 format handles this by marking both strings A1 (*This sentence* and *is self-referential*), but adding a C- prefix to the argument tag on the second argument. Another difference between the PropBank annotation and the CoNLL-2005 version of it is their treatments of filler gap constructions involving empty categories. PropBank annotation includes the whole chain of empty categories, as well as the antecedent of the empty category (the filler of the gap). In contrast, the CoNLL-2005 version only includes the filler of the gap and if there is no filler, the argument is omitted, e.g., no ARG0 (subject) for *leave* would be included in *I said to leave* because the subject of *leave* is unspecified.

3.1.4 NomBank

NomBank annotation (Meyers et al., 2004) uses essentially the same framework as PropBank to annotate arguments of nouns. Differences between PropBank and NomBank stem from differences between noun and verb argument structure; differences in treatment of nouns and verbs in the Penn Treebank; and differences in the sophistication of previous research about noun and verb argument structure. Only the subset of nouns that take arguments are annotated in NomBank and only a subset of the non-argument siblings of nouns are marked as ARGM. These limitations were necessary to make the NomBank task consistent and tractable. In addition, long distance dependencies of nouns, e.g., the relation between *Mary* and *walk* in *Mary took dozens of walks* is handled as follows: *Mary* is marked as the ARG0 of *walk* and *took + dozens + of* is marked as a support chain in NomBank. In contrast, verbal long distance dependencies can be handled by means of empty categories in the Penn Treebank, e.g., the relation be-

⁸PropBank I is used here. Later versions of PropBank mark instances of *be* in addition to other verbs. PropBank's use of the terms *roleset* and *ARGM* correspond approximately to *sense* and *adjunct* in common usage.

⁷<http://projects.ldc.upenn.edu/ace/>

tween *John* and *walked* in *John seemed t to walk*. Support chains are needed because nominal long distance dependencies are not captured under the Penn Treebank’s system of empty categories.

3.2 Conversion to Dependencies

3.2.1 Syntactic Dependencies

There exists no large-scale dependency treebank for English, and we thus had to construct a dependency-annotated corpus automatically from the Penn Treebank (Marcus et al., 1993). Since dependency syntax represents grammatical structure by means of labeled binary head–dependent relations rather than phrases, the task of the conversion procedure is to identify and label the head–dependent pairs. The idea underpinning constituent-to-dependency conversion algorithms (Magerman, 1994; Collins, 1999; Yamada and Matsumoto, 2003) is that head–dependent pairs are created from constituents by selecting one word in each phrase as the head and setting all other as its dependents. The dependency labels are then inferred from the phrase–subphrase or phrase–word relations.

Our conversion procedure (Johansson and Nugues, 2007) differs from this basic approach by exploiting the rich structure of the constituent format used in Penn Treebank 3:

- Grammatical function labels that often can be directly used in the dependency framework.
- Long-distance grammatical relations represented by means of empty categories and secondary edges, which can be used to create (often nonprojective) dependency links.

Of the grammatical function tags available in the Treebank, we removed the HLN, NOM, TPC, and TTL tags since they represent structural properties of single phrases rather than binary relations. For compatibility between the WSJ and Brown corpora, we removed the ETC, UNF, and IMP tags from Brown and the CLR tag from WSJ.

Algorithms 1 and 2 show the constituent-to-dependency conversion algorithm and function labeling, respectively. The first steps apply structural transformations to the constituent trees. Next, a head word is assigned to each constituent. After this, grammatical functions are inferred, allowing a dependency tree to be created.

To find head children (used in `assign-heads`), a system of rules is used

Algorithm 1: Pseudocode for constituent-to-dependency conversion.

```

procedure constituents-to-dependencies( $T$ )
  import-glarf( $T$ )
  reattach-traces( $T$ )
  split-small-clauses( $T$ )
  assign-heads( $T.root$ )
  assign-functions( $T$ )
  return create-dependency-tree( $T$ )

procedure import-glarf( $T$ )
  Import a GLARF surface dependency graph  $G$ 
  for each multi-word name  $N$  in  $G$ 
    for each token  $d$  in  $N$ 
      Set the function tag of  $d$  to NAME
  for each dependency link  $h \rightarrow_L d$  in  $G$ 
    if  $L \in \{ \text{APPOSITE, A-POS, N-POS, POST-HON, Q-POS, RED-RELATIVE, SUFFIX, T-POS, TITLE} \}$ 
      or if  $h$  and  $d$  are inside a split word
        Set the function tag of  $d$  to  $L$  in  $T$ 
      if  $h$  and  $d$  are part of a larger constituent
        Add an NX constituent to  $T$  that brackets  $h$  and  $d$ 

procedure reattach-traces( $T$ )
  for each empty category  $t$  in  $T$ 
    if  $t$  is linked to a constituent  $C$  via a secondary edge label  $L$ 
      and  $L \in \{ *ICH*, *T*, *RNR* \}$ 
        disconnect  $C$ 
        disconnect the secondary edge
        attach  $C$  to the parent of  $t$ 

procedure split-small-clauses( $T$ )
  for each verb phrase  $C$  in  $T$ 
    if  $C$  has a child  $S$  and the phrase label of  $S$  is  $S$ 
      and  $S$  is not preceded by a `` or , tag
      and  $S$  has a subject child  $s$ 
        disconnect  $s$ 
        attach  $s$  to  $C$ 
        set the function tag of  $s$  to OBJ
        set the function tag of  $S$  to OPRD

procedure assign-heads( $N$ )
  for each child  $C$  of  $N$ 
    assign-heads( $C$ )
  if is-coordinated( $N$ )
     $e \leftarrow$  index of first CC or CONJP or , or :
  else
     $e \leftarrow$  index of last child of  $N$ 
  find head child  $H$  between 1 and  $e$  according to head rules (Table 3)
   $N.head \leftarrow H.head$ 

procedure is-coordinated( $N$ )
  if  $N$  has the label UCP return True
  if  $N$  has a CC or CONJP child which is not leftmost return True
  if  $N$  has a , or : child  $c$ , and  $c$  is not leftmost or rightmost or
  crossed by an apposition link, return True
  else return False

procedure create-dependency-tree( $T$ )
   $D \leftarrow \{ \}$ 
  for each token  $t$  in  $T$ 
    let  $C$  be the highest constituent that  $t$  is the head of
    let  $P$  be the parent of  $C$ 
    let  $L$  be the function tag of  $C$ 
     $D \leftarrow D \cup P.head \rightarrow_L t$ 
  return  $D$ 

```

(Table 3). The first column in the table indicates the phrase type, the second is the search direction, and the third is a priority list of phrase types to look for. For instance, to find the head of an S phrase, we look from right to left for a VP . If no VP is found, look for anything with a PRD function tag, and so on.

Moreover, since the grammatical structure in-

ADJP	←	NNS QP NN \$ ADVP JJ VBN VBG ADJP JJR NP JJS DT FW RBR RBS SBAR RB
ADVP	→	RB RBR RBS FW ADVP TO CD JJR JJ IN NP JJS NN
CONJP	→	CC RB IN
FRAG	→	(NN* NP) W* SBAR (PP IN) (ADJP JJ) ADVP RB
INTJ	←	*
LST	→	LS :
NAC, NP, NX, WHNP	←	(NN* NX) NP-ε JJR CD JJ JJS RB QP NP
PP, WHPP	→	IN TO VBG VBN RP FW
PRN	→	S* N* W* PP IN ADJP JJ* ADVP RB*
PRT	→	RP
QP	←	\$ IN NNS NN JJ RB DT CD NCD QP JJR JJS
RRC	→	VP NP ADVP ADJP PP
S	←	VP *-PRD S SBAR ADJP UCP NP
SBAR	←	S SQ SINV SBAR FRAG IN DT
SBARQ	←	SQ S SINV SBARQ FRAG
SINV	←	VBZ VBD VBP VB MD VP *-PRD S SINV ADJP NP
SQ	←	VBZ VBD VBP VB MD *-PRD VP SQ
UCP	→	*
VP	→	VBD VBN MD VBZ VB VBG VBP VP *-PRD ADJP NN NNS NP
WHADJP	←	CC WRB JJ ADJP
WHADVP	→	CC WRB
X	→	*

Table 3: Head rules.

Algorithm 2: Pseudocode for the function labeling procedure.

```

procedure assign-functions( $T$ )
  for each constituent  $C$  in  $T$ 
    if  $C$  has no function tag from Penn or GLARF
       $L \leftarrow$  infer-function( $C$ )
      Set the function tag of  $C$  to  $L$ 

procedure infer-function( $C$ )
  let  $e$  be the head of  $C$ ,  $P$  the parent of  $C$ , and  $p$  the head of  $P$ 
  if  $C$  is an object return OBJ
  if  $C$  is PRN return PRN
  if  $h$  is punctuation return P
  if  $C$  is coordinated with  $P$  return COORD
  if  $C$  is PP, ADVP, or SBAR and  $P$  is VP return ADV
  if  $C$  is PRT and  $P$  is VP return PRT
  if  $C$  is VP and  $P$  is VP, SQ, or SINV return VC
  if  $C$  is TO and  $P$  is VP return IM
  if  $P$  is SBAR and  $p$  is IN return SUB
  if  $P$  is VP, S, SBAR, SBARQ, SINV, or SQ and  $C$  is RB return ADV
  if  $P$  is NP, NX, NAC, or WHNP return NMOD
  if  $P$  is ADJP, ADVP, WHADJP, or WHADVP return AMOD
  if  $P$  is PP or WHPP return PMOD
  else return DEP

```

side noun phrases (NP) is under-specified in the Penn Treebank, we imported dependencies inside NPs and hyphenated words from a version of the Penn Treebank mapped into GLARF, the Grammatical and Logical Argument Representation Framework (Meyers et al., 2007).

The parts of GLARF’s NP analysis that are most relevant to this task include: (i) identifying appositives (APPO, e.g., that *book* depends on *gift* in *Mary’s gift, a book about cheese*; (ii) the identification of name boundaries taken from BBN’s

NE annotation, e.g., identifying that *Smith* depends on *Mary* which depends on *appointment* in *the Mary Smith appointment*; (iii) identifying TITLE and POSTHON dependencies, e.g., determining that *Ms.* and *III* depend on *Mary* in *Ms. Mary Smith III*. These identifications were carried out by hand-coded rules that have been fine tuned as part of GLARF, over the past several years. For example, identifying apposition constructions requires identifying that both the head and the apposite can stand alone – proper nouns (*John Smith*), plural nouns (*books*), and singular common nouns with determiners (*the book*) are stand-alone cases, whereas singular nouns without determiners (*green book*) do not qualify.

We split Treebank tokens at a hyphen (-) or a forward slash (/) if the segments on either side of these delimiters are: (a) a word in a dictionary (COMLEX Syntax or any of the dictionaries available on the NOMLEX website); (b) part of a markable Named Entity;⁹ or (c) a prefix from the list: *co, pre, post, un, anti, ante, ex, extra, fore, non, over, pro, re, super, sub, tri, bi, uni, ultra*. For example, *York-based* was split into 3 segments: (1) *York*, (2) - and (3) *based*.

⁹The CoNLL-2008 website contains a *Named Entity Token gazetteer* to aid in this segmentation.

3.2.2 Semantic Dependencies

When encoding the semantic dependencies, it was necessary to convert the underlying constituent analysis of PropBank and NomBank into a dependency analysis. Because semantic predicates are already assigned to individual tokens in both PropBank (the version used for the CoNLL-2005 shared task) and NomBank, constituent-to-dependency conversion is thus necessary only for semantic arguments. Conceptually, this conversion can be handled using similar heuristics as described in Section 3.2.1. However, in order to avoid replicating this effort and to ensure compatibility between syntactic and semantic dependencies, we decided to generate semantic dependencies using only argument boundaries and the syntactic dependencies generated in Section 3.2.1, i.e., ignoring syntactic constituents. Given this input, we identify the head of a semantic argument using the following heuristic:

The head of a semantic argument is assigned to the token inside the argument boundaries whose head is a token outside the argument boundaries.

This heuristic works remarkably well: over 99% of the PropBank arguments in the training corpus have a single token whose head is located outside of the argument boundaries. As a simple example, consider the following annotated text: *[sold]*_{PRED} *[1214 cars]*_{ARG1} *[in the U.S.]*_{ARGM-LOC}. Using the above heuristic, the head of the ARG1 argument is set to *cars*, because it has an OBJ dependency to *sold*, and the head of the ARGM-LOC argument is set to *in*, because it modifies *sold* through a LOC dependency.

While this heuristic processes the vast majority of arguments, there are several cases that require special treatment. We discuss these situations in the remainder of this section.

Arguments with several syntactic heads

For 0.7% of the semantic arguments, the above heuristic detects several syntactic heads for the given boundary. For example, in the text *[it]*_{ARG0} *[expects]*_{PRED} *[its U.S. sales to remain steady at about 1200 cars]*_{ARG1}, the above heuristic assigns two syntactic heads to ARG1: *sales*, which modifies *expects* through an OBJ dependency, and *to*, which modifies *expects* through a PRD dependency. These situations are caused

by the constituent-to-dependency conversion process described in Section 3.2.1, which in some cases interprets syntax differently than the original Treebank annotation, e.g., the raising phenomenon for the PRD dependency in the above example. In such cases, we split the original argument into a sequence of discontinuous arguments, e.g., the ARG1 in the above example becomes *[its U.S. sales]*_{ARG1} *[to remain steady at about 1200 cars]*_{C-ARG1}.

Merging discontinuous arguments

While in the above case we split arguments, there are situations where we can merge arguments that were initially discontinuous in PropBank or NomBank. This typically happens when the PropBank/NomBank predicate is infixed inside one of its arguments. For example, in the text *[Million-dollar conferences]*_{ARG1} *were* *[held]*_{PRED} *[to chew on subjects such as...]*_{C-ARG1}, PropBank lists multiple constituents as aggregately filling the ARG1 slot of *held*. These cases are detected automatically because the least common ancestor of the argument pieces is actually one of the argument segments. In the above example, *to chew on subjects such as...* depends on *Million-dollar conferences* because *to* modifies *conferences* through a NMOD dependency. In these situations, we treat the least common ancestor, e.g., *conferences* in the above text, as the true argument. This heuristic allowed us to merge 1665 (or 0.6% of total) arguments that were initially discontinuous in the PropBank training corpus.

Empty categories

PropBank and NomBank both encode chains of empty categories. As with the 2005 shared task (Carreras and Màrquez, 2005), we used the head of the antecedent of empty categories as arguments rather than empty categories. Furthermore, empty category arguments with no antecedents were ignored.¹⁰ For example, given *The man wanted t to make a speech*, we assume that the A0 of *make* and *speech* is *man*, rather than the chain consisting of the empty category represented as *t* and *man*.

Annotation disagreements

NomBank and Penn Treebank annotators sometimes disagree about constituent structure. Nom-

¹⁰Under our approach to filler gap constructions, the filler is a shared argument (as in Relational Grammar, most Feature Structure and Dependency Grammar frameworks), in contrast with the Penn Treebank's empty category antecedent approach (more closely resembling the various Chomskian approaches).

Label	Freq.	Description
NMOD	324834	Modifier of nominal
P	135260	Punctuation
PMOD	115988	Modifier of preposition
SBJ	89371	Subject
OBJ	66677	Object
ROOT	49178	Root
ADV	47379	General adverbial
NAME	41138	Name-internal link
VC	35250	Verb chain
COORD	31140	Coordination
DEP	29456	Unclassified
TMP	26305	Temporal adverbial or nominal modifier
CONJ	24522	Second conjunct (dependent on conjunction)
LOC	18500	Locative adverbial or nominal modifier
AMOD	17868	Modifier of adjective or adverbial
PRD	16265	Predicative complement
APPO	16163	Apposition
IM	16071	Infinitive verb (dependent on infinitive marker <i>to</i>)
HYPH	14073	Token part of a hyphenated word (dependent on a preceding part of the hyphenated word)
HMOD	13885	Token inside a hyphenated word (dependent on the head of the hyphenated word)
SUB	12995	Subordinated clause (dependent on subordinating conjunction)
OPRD	11707	Predicative complement of raising/control verb
SUFFIX	10548	Possessive suffix (dependent on possessor)
DIR	6145	Adverbial of direction
TITLE	5917	Title (dependent on name)
MNR	4753	Adverbial of manner
POSTHON	4377	Posthonorific modifier of nominal
PRP	4013	Adverbial of purpose or reason
PRT	3235	Particle (dependent on verb)
LGS	3115	Logical subject of a passive verb
EXT	2374	Adverbial of extent
PRN	2176	Parenthetical
EXTR	658	Extraposited element in cleft
DTV	496	Dative complement (<i>to</i>) in dative shift
PUT	271	Complement of the verb <i>put</i>
BNF	44	Benefactor complement (<i>for</i>) in dative shift
VOC	24	Vocative

Table 4: Statistics for atomic syntactic labels.

Bank annotators are in effect assuming that the constituents provided form a phrase. In this case, the constituents are adjacent to each other. For example, consider the NP *the human rights discussion*. In this case, the Penn Treebank would treat each of the four words *the*, *human*, *rights*, *discussion* as daughters of a single NP node. However, NomBank would treat *human rights* as a single ARG1 of *discussion*. Since noun noun modification constructions are head final, we can easily determine (via GLARF) that *rights* is the markable dependent of *discussion*.

Support chains

Finally, NomBank’s encoding of support chains is handled as chains of dependencies in the data (although these are not scored). For example, given *Mary took dozens of walks*, where *Mary* is the ARG0 of *walks*, the support chain *took + dozens + of* is represented as a sequence of dependencies: *of* depends on *Mary*, *dozens* depends on *of* and *took*

depends on *dozens*. Each of these dependencies is labeled *SU*.

3.3 Overview of Corpora

The syntactic dependency types are divided into *atomic* types that consist of a single label, and *non-atomic* types consisting of more than one label. There are 38 atomic and 70 non-atomic labels in the corpus. There are three types of non-atomic labels: those consisting of a PRD or OPRD concatenated with an adverbial label such as LOC or TMP; gapping labels such as GAP-SBJ; and combined adverbial tags such as LOC-TMP.

Table 4 shows statistics for the atomic syntactic dependencies: label type, the frequency of the label in the complete corpus, and a description of the label. Table 5 shows the corresponding statistics for non-atomic dependencies, excluding gapping dependencies. The non-atomic labels are rare, which made it difficult to learn these relations ef-

Label	Frequency
LOC-PRD	798
PRD-TMP	51
PRD-PRP	45
LOC-OPRD	31
DIR-PRD	4
MNR-PRD	3
LOC-TMP	2
MNR-TMP	1
LOC-MNR	1
DIR-OPRD	1

Table 5: Statistics for non-atomic syntactic labels excluding gapping labels.

Label	Frequency
GAP-SBJ	116
GAP-OBJ	102
DEP-GAP	83
GAP-TMP	69
GAP-PRD	66
GAP-LGS	44
GAP-LOC	42
DIR-GAP	37
GAP-PMOD	22
GAP-VC	20
EXT-GAP	16
ADV-GAP	15
GAP-NMOD	13
GAP-LOC-PRD	6
DTV-GAP	6
AMOD-GAP	6
GAP-MNR	5
GAP-PRP	4
EXTR-GAP	3
GAP-SUB	1
GAP-PUT	1
GAP-OPRD	1

Table 6: Statistics for non-atomic labels containing a gapping label.

fectively. Table 6 shows the table for non-atomic labels containing a gapping label.

A dependency link $w_i \rightarrow w_j$ is said to be *projective* if all words occurring between w_i and w_j in the surface word order are dominated by w_i (where dominance is the transitive closure of the direct link relation). Nonprojective links are impossible to handle for the search procedures in many types of dependency parsers. It has been previously observed that the majority of dependencies in all languages are projective, and this is particularly true for English – in the complete corpus, only 4118 links (0.4%) are nonprojective. 3312 sentences, or 7.6%, contain at least one nonprojective link.

Table 7 shows statistics for different types of nonprojective links: nonprojectivity caused by *wh-movement*, such as in *Where are you going?* or *What have you done?*; split clauses such as

Type	Frequency
<i>wh-movement</i>	1709
Split clause	734
Split noun phrase	590
Other	1085

Table 7: Statistics for nonprojective links.

POS	Frequency
NN	68477
NNS	30048
VBD	24106
VB	23650
VCN	19339
VBG	14245
VBZ	10883
VBP	6330
Other	83

Table 8: Statistics for predicates, by POS tags.

Even to make love, he says, you need experience; split noun phrases such as *hold a hearing tomorrow on the topic*; and all other types of nonprojective links.

Lastly, Tables 8 and 9 summarizes statistics for semantic predicates and roles. Table 8 shows the number of non-support predicates with a given POS tag in the whole corpus (we used GPOS or PPOSS for predicates inside hyphenated words). The last line shows the number of predicates with a POS tag that does not start with NN or VB. This last table entry is generated by POS tagger mistakes when producing the PPOSS tags, or by errors in our NomBank/PropBank conversion software.¹¹ Nevertheless, the overall picture given by the table indicates that predicates are almost perfectly distributed between nouns and verbs: there are 98525 nominal and 98553 verbal predicates.

Table 9 shows the number of arguments with a given role label. For brevity we list only labels that are instantiated at least 10 times in the whole corpus. The total number of arguments labeled with a role label with frequency lower than 10 is listed in the last line in the table. The table indicates that, while the top three most common role labels are “core” labels (A1, A0, A2), modifier arguments (AM-*) account for approximately 20% of the total number of arguments. On the other hand, discontinuous arguments are not common: only 0.7% of the total number of arguments have a continuation label (C-*).

¹¹In very few situations, we select incorrect head tokens for multi-word predicates.

Label	Frequency
A1	161409
A0	109437
A2	51197
AM-TMP	25913
AM-MNR	13080
AM-LOC	11409
A3	10269
AM-MOD	9986
AM-ADV	9496
AM-DIS	5369
R-A0	4432
AM-NEG	4097
A4	3281
C-A1	3118
R-A1	2565
AM-PNC	2445
AM-EXT	1428
AM-CAU	1346
AM-DIR	1318
R-AM-TMP	797
R-A2	307
R-AM-LOC	246
R-AM-MNR	155
A5	91
AM-PRD	78
C-A0	70
C-A2	65
R-AM-CAU	50
C-A3	37
R-A3	29
C-AM-MNR	24
C-AM-ADV	20
AM-REC	16
AA	14
R-AM-PNC	12
C-AM-EXT	11
C-AM-TMP	11
C-A4	11
Frequency < 10	70

Table 9: Statistics for semantic roles.

4 Submissions and Results

Nineteen groups submitted test runs in the closed challenge and five groups participated in the open challenge. Three of the latter groups participated only in the open challenge, and two of these submitted results only for the semantic subtask. These results are summarized in Tables 10 and 11.

Table 10 summarizes the official results – i.e., results at evaluation deadline – for the closed challenge. Note that several teams corrected bugs and/or improved their systems and they submitted post-evaluation scores (accounted in the shared task website). The table indicates that most of the top results cluster together: three systems had a labeled macro F_1 score on the WSJ+Brown corpus around 82 points (che, ciaramita, and zhao); five systems scored around 79 labeled macro F_1 points (yuret, samuelsson, zhang, henderson, and

watanabe). Remarkably, the top-scoring system (johansson) is in a class of its own, with scores 2–3 points higher than the next system. This is most likely caused by the fact that Johansson and Nugues (2008) implemented a thorough system that addressed all facets of the task with state-of-the-art methods: second-order parsing model, argument identification/classification models separately tuned for PropBank and NomBank, reranking inference for the SRL task, and, finally, joint optimization of the complete task using meta-learning (more details in Section 5).

Table 11 lists the official results in the open challenge. The results in this challenge are lower than in the closed challenge, but this was somewhat to be expected considering that there were fewer participants in this challenge and none of the top five groups in the closed challenge submitted results in the open challenge. Only one of the systems that participated in both challenges (zhang) improved the results submitted in the closed challenge. Zhang et al. (2008) achieved this by extracting features for their semantic subtask models both from the parser used in the closed challenge and a secondary parser that was trained on a different corpus. The improvements measured were relatively small for the in-domain WSJ corpus (0.2 labeled macro F_1 points) but larger for the out-of-domain Brown corpus (approximately 1 labeled macro F_1 point).

Tables 10 and 11 indicate that in both challenges the results on the out-of-domain corpus (Brown) are much lower than the results measured in-domain (WSJ). The difference is around 7–8 LAS points for the syntactic subtask and 12–14 labeled F_1 points for semantic dependencies. Overall, this yields a drop of approximately 10 labeled macro F_1 points for most systems. This performance decrease on out-of-domain corpora is consistent with the results reported in CoNLL-2005 on SRL (using the same Brown corpus). These results indicate that domain adaptation is a problem that is far from being solved for both syntactic and semantic analysis of text. Furthermore, as the scores on the syntactic and semantic subtasks indicate, domain adaptation becomes even harder as the task to be solved gets more complex.

We describe the participating systems in the next section. Then, in Section 6, we revert to result analysis using different evaluation measures and different views of the data.

	Labeled Macro F ₁ (complete task)			Labeled Attachment Score (syntactic dependencies)			Labeled F ₁ (semantic dependencies)		
	WSJ+Brown	WSJ	Brown	WSJ+Brown	WSJ	Brown	WSJ+Brown	WSJ	Brown
johansson	84.86 (1)	85.95	75.95	89.32 (1)	90.13	82.81	80.37 (1)	81.75	69.06
che	82.66 (2)	83.78	73.57	86.75 (5)	87.51	80.73	78.52 (2)	80.00	66.37
ciaramita	82.06 (3)	83.25	72.46	86.60 (11)	87.47	79.67	77.50 (3)	79.00	65.24
zhao	81.44 (4)	82.62	71.78	86.66 (8)	87.52	79.83	76.16 (4)	77.67	63.69
yuret	79.84 (5)	80.97	70.55	86.62 (10)	87.39	80.46	73.06 (5)	74.54	60.62
samuelsson	79.79 (6)	80.92	70.49	86.63 (9)	87.36	80.77	72.94 (6)	74.47	60.18
zhang	79.32 (7)	80.41	70.48	87.32 (2)	88.14	80.80	71.31 (7)	72.67	60.16
henderson	79.11 (8)	80.19	70.34	86.91 (4)	87.78	80.01	70.97 (8)	72.26	60.38
watanabe	79.10 (9)	80.30	69.29	87.18 (3)	88.06	80.17	70.84 (9)	72.37	58.21
morante	78.43 (10)	79.52	69.55	86.07 (12)	86.88	79.58	70.51 (10)	71.88	59.23
li	78.35 (11)	79.38	70.01	86.69 (6)	87.42	80.8	69.95 (11)	71.27	59.17
<i>baldridge</i>	77.49 (12)	78.57	68.53	86.67 (7)	87.42	80.64	67.92 (14)	69.35	55.95
chen	77.00 (13)	77.95	69.23	84.47 (16)	85.20	78.58	69.45 (12)	70.62	59.81
lee	76.90 (14)	77.96	68.34	84.82 (15)	85.69	77.83	68.71 (13)	69.95	58.63
sun	76.28 (15)	77.10	69.58	85.75 (13)	86.37	80.75	66.61 (15)	67.62	58.26
<i>choi</i>	71.23 (16)	72.22	63.44	77.56 (17)	78.58	69.46	64.78 (16)	65.72	57.4
<i>trandabat</i>	63.45 (17)	64.21	57.41	85.21 (14)	85.96	79.24	40.63 (17)	41.36	34.75
lluis	63.29 (18)	63.74	59.65	71.95 (18)	72.30	69.14	54.52 (18)	55.09	49.95
neumann	19.93 (19)	20.13	18.14	16.25 (19)	16.22	16.47	22.36 (19)	22.86	17.94

Table 10: Official results in the closed challenge (post-evaluation scores are available on the shared task website). Teams are denoted by the last name of the first author of the corresponding paper in the proceedings or the last name of the person who registered the team if no paper was submitted. Italics indicate that there is no corresponding paper in the proceedings. Results are sorted in descending order of the labeled macro F₁ score on the WSJ+Brown corpus. The number in parentheses next to the WSJ+Brown scores indicates the system rank in the corresponding task.

	Labeled Macro F ₁ (complete task)			Labeled Attachment Score (syntactic dependencies)			Labeled F ₁ (semantic dependencies)		
	WSJ+Brown	WSJ	Brown	WSJ+Brown	WSJ	Brown	WSJ+Brown	WSJ	Brown
vickrey	–	–	–	–	–	–	76.17 (1)	77.38	66.23
riedel	–	–	–	–	–	–	74.59 (2)	75.72	65.38
zhang	79.61 (1)	80.61	71.45	87.32 (1)	88.14	80.80	71.89 (3)	73.08	62.11
li	77.84 (2)	78.87	69.51	86.69 (2)	87.42	80.80	68.99 (4)	70.32	58.22
wang	76.19 (3)	78.39	59.89	84.56 (3)	85.50	77.06	67.12 (5)	70.41	42.67

Table 11: Official results in the open challenge (post-evaluation scores are available on the shared task website). Teams are denoted by the last name of the first author of the corresponding paper in the proceedings or the last name of the person who registered the team if no paper was submitted. Italics indicate that there is no corresponding paper in the proceedings. Results are sorted in descending order of the labeled F₁ score for semantic dependencies on the WSJ+Brown corpus. The number in parentheses next to the WSJ+Brown scores indicates the system rank in the corresponding task.

5 Approaches

Table 5 summarizes the properties of the systems that participated in the closed the open challenges. The second column of the table highlights the overall architectures. We used + to indicate that the components are sequentially connected. The lack of a + sign indicates that the corresponding tasks are performed jointly. For example, Riedel and Meza-Ruiz (2008) perform predicate and argument identification and classification jointly, whereas Ciaramita et al. (2008) implemented a pipeline architecture of three components. We use the || to indicate that several differ-

ent architectures that span multiple subtasks were deployed in parallel.

This summary of system architectures indicates that it is common that systems combine several components in the semantic or syntactic subtasks – e.g., nine systems jointly performed predicate/argument identification and classification – but only four systems combined components between the syntactic and semantic subtasks: Henderson et al. (2008), who implemented a generative history-based model (Incremental Sigmoid Belief Networks with vectors of latent variables) where syntactic and semantic structures are separately

generated but using a synchronized derivation (sequence of actions); Samuelsson et al. (2008), who, within an ensemble-based architecture, implemented a joint syntactic-semantic model using MaltParser with labels enriched with semantic information; Lluís and Màrquez, who used a modified version of the Eisner algorithm to jointly predict syntactic and semantic dependencies; and finally, Sun et al. (2008), who integrated dependency label classification and argument identification using a maximum-entropy Markov model. Additionally, Johansson and Nugues (2008), who had the highest ranked system in the closed challenge, integrate syntactic and semantic analysis in a final reranking step, which maximizes the joint syntactic-semantic score in the top k solutions. In the same spirit, Chen et al. (2008) search in the top k solutions for the one that maximizes a global measure, in this case the joint probability of the complete problem. These joint learning strategies are summarized in the **Joint Learning/Opt.** column in the table. The system of Riedel and Meza-Ruiz (2008) deserves a special mention: even though Riedel and Meza-Ruiz did not implement a syntactic parser, they are the only group that performed the complete SRL subtask – i.e., predicate identification and classification, argument identification and classification – jointly, simultaneously for all the predicates in a sentence. They implemented a joint SRL model using Markov Logic Networks and they selected the overall best solution using inference based on the cutting-plane algorithm.

Although some of the systems that implemented joint approaches obtained good results, the top five systems in the closed challenge are essentially systems with pipeline architectures. Furthermore, Johansson and Nugues (2008) and Riedel and Meza-Ruiz (2008) showed that joint learning/optimization improves the overall results, but the improvement is not large. These initial efforts indicate at least that the joint modeling of this problem is not a trivial task.

The **D Arch.** and **D Inference** columns summarize the parsing architectures and the corresponding inference strategies. Similar to last year’s shared task (Nivre et al., 2007), the vast majority of parsing models fall in two classes: transition-based (“trans” in the table) or graph-based (“graph”) models. By and large, transition-based models use a greedy inference strategy, whereas graph-based

models used different Maximum Spanning Tree (MST) algorithms: Carreras (2007) – MST^C , Eisner (2000) – MST^E , or Chu-Liu/Edmonds (McDonald et al., 2005; Chu and Liu, 1965; Edmonds, 1967) – $MST^{CL/E}$. More interestingly, most of the best systems used some strategy to mitigate parsing errors. In the top three systems in the closed challenge, two (che and ciaramita) used parser combination through voting and/or stacking of different models (see the **D Comb.** column). Samuelsson et al. (2008) perform a MST inference with the bag of all dependencies output by the individual MALT parser variants. Johansson and Nugues (2008) use a single parsing model, but this model is extended with second-order features.

The **PA Arch.** and **PA Inference** columns summarize the architectures and inference strategies used for the identification and classification of predicates and arguments. The columns indicate that most systems modeled the SRL problem as a token-by-token classification problem (“class” in the table) with a corresponding greedy inference strategy. Some systems (e.g., yuret, samuelsson, henderson, lluis) incorporate SRL within parsing, in which case we report the corresponding parsing architecture and inference approach. Vickrey and Koller (2008) simplify the sentences to be labeled using a set of hand-crafted rules before deploying a classification model on top of a constituent-based representation. Unlike in the case of parsing, few systems (yuret, samuelsson, and morante) combine several PA models and the combination is limited to simple voting strategies (see the **PA Comb.** column).

Finally, the **ML Methods** column lists the Machine Learning (ML) methods used. The column indicates that maximum entropy (ME) was the most popular method (12 distinct systems relied on it). Support Vector Machines (SVM) (eight systems) and the Perceptron algorithm (three systems) were also popular ML methods.

6 Analysis

Section 4 summarized the results in the closed and open challenges using the official evaluation measures. In this section, we analyze the submitted runs using different evaluation measures, e.g., Exact Match or Perfect Proposition F_1 scores, and different views of the data, e.g., only non-projective dependencies or NomBank versus PropBank frames.

closed	Overall Arch.	D Arch.	D Comb.	D Inference	PA Arch.	PA Comb.	PA Inference	Joint Learning/Opt.	ML Methods
johansson	D+PI+PC+AI+AC	graph	no	MST ^C	class	no	rerank	rerank	Perceptron, ME
che	D+PI+PC+AI+AC	graph	stacking	MST ^{C/L/E}	class	no	ILP	no	ME
ciaramita	D+PIC+AI+AC	trans	voting, stacking	greedy	class	no	rerank	no	SVM, ME, Perceptron
zhao	D+AI+AC+PI+PC	trans	no	greedy	class	no	greedy	no	ME
yuret	D+(PIC+AI+AC PIC+AI+AC)	graph	no	MST ^E	class, generative	voting	greedy	no	MLE, MBL
samuelsson	D+PI+ (AI+AC DAIC)+PC	trans	MST ^{C/L/E} blending	greedy	class, trans	voting	greedy	unified labels	SVM
zhang	D+PI+AI+AC+PC	graph, trans	meta-learning	MST ^{C/L/E} , greedy	class	no	greedy	no	SVM, ME
henderson	DPAIC+D	generative, trans	no	beam search	trans	no	beam search	synchronized derivation	ISBN
watanabe	DI+DC+PI+PC+AI+AC	relative preference model	no	greedy tournament model, Viterbi	class	no	no	no	SVM, CRF, MBL
morante	D+PI+AI+AC	trans	no	greedy	class	voting	greedy	no	SVM, MBL
li	D+PIC+AI+AC	graph	no	MST ^{C/L/E}	class	voting	greedy	no	ME
chen	D+PI+PC+AI+AC	trans	no	prob	class	no	prob	global probability optimization	ME
lee	D+PI+AI+AC+PC	trans	no	greedy	class	no	prob	no	SVM, ME
sun	DI+PI+DCAI+AC	graph	no	MST ^E , Viterbi	graph	no	Viterbi, ILP	MEMM, Viterbi	ME
lluis	D+PI+DAIC+PC	graph	no	MST ^E	graph	no	MST ^E	MST ^E	Perceptron, SVM
neumann	D+PI+PC+AI+AC	trans	no	greedy	class	no	no	no	ME
open									
vickrey	AI+AC+PI+PC	-	-	-	sentence simplification, class	no	greedy	-	ME
riedel	PAIC	-	-	-	Markov Logic Network	no	Cutting Plane	-	MIRA
wang	PI+AI+AC	trans, graph	no	greedy, MST ^{C/L/E}	class	no	prob	no	SVM, ME, MIRA

Table 12: Summary of system architectures that participated in the closed and open challenges. The closed-challenge systems are sorted by macro labeled F₁ score on the WSJ+Brown corpus. Because some open-challenge systems did not implement syntactic parsing, these systems are sorted by labeled F₁ score of the semantic dependencies on the WSJ+Brown corpus. Only the systems that have a corresponding paper in the proceedings are included. Systems that participated in both challenges are listed only in the closed challenge. Acronyms used: **D** - syntactic dependencies, **P** - predicate, **A** - argument, **I** - identification, **C** - classification. **Overall arch.** stands for the complete system architecture; **D Arch.** stands for the architecture of the syntactic parser; **D Comb.** indicates if the final parser output was generated using parser combination; **D Inference** stands for the type of inference used for syntactic parsing; **PA Arch.** stands for the architecture used for PAIC; **PA Comb.** indicates if the PA output was generated through system combination; **PA Inference** stands for the type of inference used for PAIC; **Joint Learning/Opt.** indicates if some form of joint learning or optimization was implemented for the syntactic + semantic global task; **ML methods** lists the ML methods used throughout the complete system.

closed	Exact Match (complete task)			Perfect Proposition F ₁ (semantic dependencies)		
	WSJ+Brown	WSJ	Brown	WSJ+Brown	WSJ	Brown
johansson	12.46 (1)	12.46	12.68	54.12 (1)	56.12	36.90
che	10.37 (2)	10.21	11.50	48.05 (2)	50.15	30.90
ciaramita	9.27 (3)	9.04	10.80	46.05 (3)	48.05	28.61
zhao	9.20 (4)	9.00	10.56	43.19 (4)	45.23	26.14
henderson	8.11 (5)	7.75	10.33	39.24 (5)	40.64	27.51
watanabe	7.79 (6)	7.54	9.39	36.44 (6)	38.09	22.72
yuret	7.65 (7)	7.33	9.62	34.61 (9)	36.13	21.78
zhang	7.40 (8)	7.46	7.28	34.96 (8)	36.25	24.22
li	7.12 (9)	6.71	9.62	32.08 (10)	33.45	20.62
samuelsson	6.94 (10)	6.62	8.92	35.20 (7)	36.96	20.22
chen	6.83 (11)	6.46	9.15	31.02 (12)	32.08	22.14
lee	6.69 (12)	6.29	9.15	31.40 (11)	32.52	22.18
morante	6.44 (13)	6.04	8.92	30.41 (14)	31.97	17.49
sun	5.38 (14)	4.96	7.98	30.43 (13)	31.51	21.40
baldrige	5.24 (15)	4.92	7.28	25.35 (15)	26.57	15.26
choi	3.33 (16)	3.50	2.58	24.77 (16)	25.71	17.37
trandabat	3.26 (17)	3.08	4.46	6.59 (18)	6.81	4.76
lluis	2.55 (18)	1.96	6.10	16.07 (17)	16.46	13.00
neumann	0.11 (19)	0.12	0.23	0.30 (19)	0.31	0.20
open						
vickrey	–	–	–	44.94 (1)	46.68	30.28
riedel	–	–	–	42.77 (2)	44.18	31.15
zhang	8.14 (1)	8.04	8.92	35.46 (3)	36.74	24.84
li	6.90 (2)	6.46	9.62	29.91 (4)	31.30	18.41
wang	5.17 (3)	5.12	5.63	18.63 (5)	20.31	7.09

Table 13: Exact Match and Perfect Proposition F₁ scores for runs submitted in the closed and open challenges. The closed-challenge systems are sorted in descending order of Exact Match scores on the WSJ+Brown corpus. Open-challenge submissions are sorted in descending order of the Perfect Proposition F₁ score. The number in parentheses next to the WSJ+Brown scores indicates the system rank according to the corresponding scoring measure.

6.1 Exact Match and Perfect Propositions

Table 13 lists the Exact Match and Perfect Proposition F₁ scores for test runs submitted in both challenges. Both these scores measure the capacity of a system to correctly parse structures with granularity much larger than a simple dependency, i.e., entire sentences for Exact Match and complete propositions for Perfect Proposition F₁ (see Section 2.2.2 for a formal definition of these evaluation measures). The table indicates that these values are much smaller than the scores previously reported, e.g., labeled macro F₁. This is to be expected: the probability of an incorrectly parsed unit (sentence or proposition) is much larger given its granularity. However, the main purpose of this analysis is to investigate if systems that focused on joint learning or optimization performed better than others with respect to these global measures. This indeed seems to be the case for at least two systems. The system of Johansson and Nugues (2008), which jointly optimizes the labeled F₁ score (for semantic dependencies) and then the labeled macro F₁ score (for the complete

task), increases its distance from the next ranked system: its Perfect Proposition F₁ score is over 6 points higher than the score of the second system in Table 13. The system of Henderson et al. (2008), which was designed for joint learning of the complete task, improves its rank from eighth to fifth compared to the official results (Table 10).

6.2 Nonprojectivity

Table 14 shows the unlabeled F1 scores for prediction of nonprojective syntactic dependencies. Since nonprojectivity is quite rare, many teams chose to ignore this issue. The table shows only those systems that submitted well-formed dependency trees, and whose output contained at least one nonprojective link. The small number of nonprojective links in the training set makes it hard to learn to predict such links, and this is also reflected in the figures. In general, the figures for nonprojective *wh*-movements and split clauses are higher, and they are also the most common types. Also, they are detectable by fairly simple patterns, such as the presence of a *wh*-word or a pair of commas.

System	All	<i>wh</i> -mov.	SpCl	SpNP
choi	25.43	49.49	45.47	8.72
lee	46.26	50.30	64.84	20.69
nugues	46.15	58.96	59.26	11.32
samuelssoon	24.47	38.15	0	9.83
titov	42.32	50.56	48.71	0
zhang	13.39	5.71	12.33	7.3

Table 14: Unlabeled F1-measures for nonprojective links. Results are given for all links, *wh*-movements, split clauses, and split noun phrases.

6.3 Normalized SRL Performance

Table 6.3 lists the scores for the semantic subtask measured as the ratio of the labeled F_1 score and LAS. As previously mentioned, this score estimates the performance of the SRL component independent of the performance of the syntactic parser. This analysis is not a substitute for the actual experiment where the SRL components are evaluated using correct syntactic information but, nevertheless, it indicates several interesting facts. First, the ranking of the top three systems in Table 10 changes: the system of Che et al. (2008) is now ranked first, and the system of Johansson and Nugues (2008) is second. This shows that Che et al. have a relatively stronger SRL component, whereas Johansson and Nugues developed a better parser. Second, several other systems improved their ranking compared to Table 10: e.g., chen from position thirteenth to ninth and choi from sixteenth to eighth. This indicates that these systems were penalized in the official ranking mainly due to the relative poor performance of their parsers.

Note that this experiment is relevant only for systems that implemented pipeline architectures, where the semantic components are in fact separated from the syntactic ones; this excludes the systems that blended syntax with SRL: henderson, sun, and lluis. Furthermore, systems that had significantly lower scores in syntax will receive an unreasonable boost in ranking according to this measure. Fortunately, there was only one such outlier in this evaluation (neumann), shown in gray in the table.

6.4 PropBank versus NomBank

Table 16 lists the labeled F_1 scores for semantic dependencies for two different views of the testing data sets: for propositions centered around verbal predicates, i.e., from PropBank, and for propositions centered around nominal predicates, i.e., from NomBank.

	Labeled F_1 / LAS		
	WSJ+Brown	WSJ	Brown
closed			
neumann	137.60 (1)	140.94	108.93
che	90.51 (2)	91.42	82.21
johansson	89.98 (3)	90.70	83.40
ciaramita	89.49 (4)	90.32	81.89
zhao	87.88 (5)	88.75	79.78
yuret	84.35 (6)	85.30	75.34
samuelssoon	84.20 (7)	85.24	74.51
choi	83.52 (8)	83.63	82.64
chen	82.22 (9)	82.89	76.11
morante	81.92 (10)	82.73	74.43
zhang	81.67 (11)	82.45	74.46
henderson	81.66 (12)	82.32	75.47
watanabe	81.26 (13)	82.18	72.61
lee	81.01 (14)	81.63	75.33
li	80.69 (15)	81.53	73.23
baldrige	78.37 (16)	79.33	69.38
sun	77.68 (17)	78.29	72.15
lluis	75.77 (18)	76.20	72.24
trandabat	47.68 (19)	48.12	43.85
open			
zhang	82.33	82.91	76.87
li	79.58	80.44	72.05
wang	79.38	82.35	55.37

Table 15: Ratio of the labeled F_1 score for semantic dependencies and LAS for syntactic dependencies. Systems are sorted in descending order of this ratio score on the WSJ+Brown corpus. We only show systems that participated in both the syntactic and semantic subtasks.

The table indicates that, generally, systems performed much worse on nominal predicates than on verbal predicates. This is to be expected considering that there is significant body of previous work that analyzes the SRL problem on PropBank, but minimal work for NomBank. On average, the difference between the labeled F_1 scores for verbal predicates and nominal predicates on the WSJ+Brown corpus is 7.84 points. Furthermore, the average difference between labeled F_1 scores on the Brown corpus alone is 12.36 points. This indicates that the problem of SRL for nominal predicates is more sensitive to domain changes than the equivalent problem for verbal predicates. Our conjecture is that, because there is very little syntactic structure between nominal predicates and their arguments, SRL models for nominal predicates select mainly lexical features, which are more brittle than syntactic or other non-lexicalized features.

Remarkably, there is one system (baldrige) which performed better on the WSJ+Brown for nominal predicates than verbal predicates. Unfortunately, this group did not submit a system-description paper so it is not clear what was their approach.

	Labeled F ₁ (verbal predicates)			Labeled F ₁ (nominal predicates)		
	WSJ+Brown	WSJ	Brown	WSJ+Brown	WSJ	Brown
closed						
johansson	84.45 (1)	86.37	71.87	74.32 (2)	75.42	60.13
che	80.46 (2)	82.17	69.33	75.18 (1)	76.64	56.87
ciaramita	80.15 (3)	82.09	67.62	73.17 (4)	74.42	57.69
zhao	77.67 (4)	79.40	66.38	73.28 (3)	74.69	54.81
samuelsson	76.17 (5)	78.03	64.00	68.13 (7)	69.58	49.24
yuret	75.91 (6)	77.88	63.02	68.81 (5)	69.98	53.58
zhang	74.82 (7)	76.62	63.15	65.61 (11)	66.82	50.18
li	74.36 (8)	76.14	62.92	62.61 (14)	63.76	47.09
henderson	73.80 (9)	75.40	63.36	66.26 (10)	67.44	50.73
watanabe	73.06 (10)	75.02	60.34	67.15 (8)	68.37	50.92
sun	72.97 (11)	74.45	63.50	58.68 (15)	59.73	45.75
morante	72.81 (12)	74.36	62.72	66.50 (9)	67.92	47.97
lee	72.34 (13)	74.15	60.49	62.83 (13)	63.66	52.18
chen	72.02 (14)	73.49	62.46	65.02 (12)	66.14	50.48
choi	70.00 (15)	71.28	61.71	56.16 (16)	57.19	44.05
baldrige	67.02 (16)	68.64	56.50	68.57 (6)	69.78	52.96
lluis	62.42 (17)	63.49	55.49	42.15 (17)	42.81	34.22
trandabat	42.88 (18)	43.79	37.06	37.14 (18)	37.89	27.50
neumann	22.87 (19)	23.53	18.24	21.7 (19)	22.04	17.14
open						
vickrey	78.41 (1)	79.75	69.57	71.86 (1)	73.29	53.25
riedel	77.13 (2)	78.72	66.75	70.25 (2)	71.03	60.17
zhang	75.00 (3)	76.62	64.44	66.76 (3)	67.79	53.76
li	73.74 (4)	75.57	62.05	61.24 (5)	62.38	46.36
wang	67.50 (5)	70.34	49.72	66.53 (4)	69.83	28.96

Table 16: Labeled F₁ scores for frames centered around verbal and nominal predicates. The number in parentheses next to the WSJ+Brown scores indicates the system rank in the corresponding data set.

Systems can mitigate the inherent differences between verbal and nominal predicates with different models for the two sub-problems. This was indeed the approach taken by two out of the top three systems (johansson and che). Johansson and Nugues (2008) developed different models for verbal and nominal predicates and implemented separate feature selection processes for each model. Che et al. (2008) followed the same method but they also implemented separate domain constraints for inference for the two models.

7 Conclusion

The previous four CoNLL shared tasks popularized and, without a doubt, boosted research in semantic role labeling and dependency parsing. This year’s shared task introduces a new task that essentially unifies the problems addressed in the past four years under a unique, dependency-based formalism. This novel task is attractive both from a research perspective and an application-oriented perspective:

- We believe that the proposed dependency-based representation is a better fit for many applications (e.g., Information Retrieval, Information Extraction) where it is often suffi-

cient to identify the dependency between the predicate and the head of the argument constituent rather than extracting the complete argument constituent.

- It was shown that the extraction of syntactic and semantic dependencies can be performed with state-of-the-art performance in linear time (Ciaramita et al., 2008). This can give a significant boost to the adoption of this technology in real-world applications.
- We hope that this shared task will motivate several important research directions. For example, is the dependency-based representation better for SRL than the constituent-based formalism? Does joint learning improve syntactic and semantic analysis?
- Surface (string related patterns, syntax, etc.) linguistic features can often be detected with greater reliability than deep (semantic) features. In contrast, deep features can cover more ground because they regularize across differences in surface strings. Machine learning systems can be more effective by using evidence from both deep and surface features jointly (Zhao, 2005).

Even though this shared task was more complex than the previous shared tasks, 22 different teams submitted results in at least one of the challenges. Building on this success, we hope to expand this effort in the future with evaluations on multiple languages and on larger out-of-domain corpora.

Acknowledgments

We want to thank the following people who helped us with the generation of the data sets: Jesús Giménez, for generating the predicted POS tags with his SVMTool POS tagger, and Massimiliano Ciaramita, for generating columns 1, 2 and 3 in the open-challenge corpus with his semantic tagger.

We also thank the following people who helped us with the organization of the shared task: Paola Merlo and James Henderson for the idea and the implementation of the Exact Match measure, Sebastian Riedel for his dependency visualization software,¹² Hai Zhao, for the the idea of the F₁ ratio score, and Carlos Castillo, for help with the shared task website. Last but not least, we thank the organizers of the previous four shared tasks: Sabine Buchholz, Xavier Carreras, Ryan McDonald, Amit Dubey, Johan Hall, Yuval Krymolowski, Sandra Kübler, Erwin Marsi, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. This shared task would not have been possible without their previous effort.

Mihai Surdeanu is a research fellow in the Ramón y Cajal program of the Spanish Ministry of Science and Technology. Richard Johansson was funded by the Swedish National Graduate School of Language Technology (GSLT). Adam Meyers' participation was supported by the National Science Foundation, award CNS-0551615 (*Towards a Comprehensive Linguistic Annotation of Language*) and IIS-0534700 (*Collaborative Research: Structure Alignment-based Machine Translation*). Lluís Màrquez's participation was supported by the Spanish Ministry of Education and Science, through research projects Trangram (TIN2004-07925-C03-02) and OpenMT (TIN2006-15307-C03-02).

References

X. Carreras. 2007. Experiments with a Higher-Order Projective Dependency Parser. In *Proc. of CoNLL-2007 Shared Task*.

¹²<http://code.google.com/p/whatswrong/>

- X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proc. of CoNLL-2005*.
- X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proc. of CoNLL-2004*.
- W. Che, Z. Li, Y. Hu, Y. Li, B. Qin, T. Liu and S. Li. 2008. A Cascaded Syntactic and Semantic Dependency Parsing System. In *Proc. of CoNLL-2008 Shared Task*.
- E. Chen, L. Shi and D. Hu. 2008. Probabilistic Model for Syntactic and Semantic Dependency Parsing. In *Proc. of CoNLL-2008 Shared Task*.
- Chinchor, N. and P. Robinson. 1998. MUC-7 Named Entity Task Definition. In *Proc. of Seventh Message Understanding Conference (MUC-7)*. http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_toc.html.
- Chomsky, Noam. 1981. *Lectures on Government and Binding*. Foris Publications, Dordrecht.
- Y.J. Chu and T.H. Liu. 1965. On the Shortest Arborescence of a Directed Graph. In *Science Sinica*, 14:1396-1400.
- M. Ciaramita, G. Attardi, F. Dell'Orletta, and M. Surdeanu. 2008. DeSRL: A Linear-Time Semantic Role Labeling System. In *Proc. of CoNLL-2008 Shared Task*.
- M. Ciaramita and Y. Altun. 2006. Broad Coverage Sense Disambiguation and Information Extraction with a Supersense Sequence Tagger. In *Proc. of EMNLP*.
- M. Collins. 1999. Head-Driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania.
- J. Edmonds. 1967. Optimum Branchings. In *Journal of Research of the National Bureau of Standards*, 71B:233-240.
- J. Eisner. 2000. Bilexical Grammars and Their Cubic-Time Parsing Algorithms. *New Developments in Parsing Algorithms*, Kluwer Academic Publishers.
- W. N. Francis and H. Kuçera. 1964. Brown Corpus. Manual of Information to accompany A Standard Corpus of Present-Day Edited American English, for use with Digital Computers. Revised 1971, Revised and Amplified 1979.
- C. Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press.
- J. Giménez and L. Màrquez. 2004. SVMTool: A general POS tagger generator based on Support Vector Machines. In *Proc. of LREC*.
- K. Hacioglu. 2004. Semantic Role Labeling Using Dependency Trees. In *Proc. of COLING-2004*.

- J. Henderson, P. Merlo, G. Musillo and I. Titov. 2008. A Latent Variable Model of Synchronous Parsing for Syntactic and Semantic Dependencies. In *Proc. of CoNLL-2008 Shared Task*.
- R. Johansson and P. Nugues. 2008. Dependency-based Syntactic–Semantic Analysis with PropBank and NomBank. In *Proc. of CoNLL-2008 Shared Task*.
- R. Johansson and P. Nugues. 2007. Extended Constituent-to-Dependency Conversion for English. In *Proc. of NODALIDA*.
- X. Lluís and L. Màrquez. 2008. A Joint Model for Parsing Syntactic and Semantic Dependencies. In *Proc. of CoNLL-2008 Shared Task*.
- D. Magerman. 1994. Natural Language Parsing as Statistical Pattern Recognition. Ph.D. thesis, Stanford University.
- M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 19.
- R. McDonald, F. Pereira, K. Ribarov and J. Hajic. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms In *Proc. of HLT-EMNLP*.
- A. Meyers, R. Grishman, M. Kosaka, and S. Zhao. 2001. Covering Treebanks with GLARF. In *Proc. of the ACL/EACL 2001 Workshop on Sharing Tools and Resources for Research and Education*.
- Meyers, A., R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The NomBank Project: An Interim Report. In *NAACL/HLT 2004 Workshop Frontiers in Corpus Annotation*, Boston.
- J. Nivre, J. Hall, J. Nilsson and G. Eryigit. 2006. Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines. In *Proc. of CoNLL-X Shared Task*.
- J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, D. Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proc. of CoNLL-2007*.
- J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007b. Malt-Parser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.
- M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1).
- S. Riedel and I. Meza-Ruiz. 2008. Collective Semantic Role Labelling with Markov Logic. In *Proc. of CoNLL-2008 Shared Task*.
- Y. Samuelsson, O. Täckström, S. Velupillai, J. Eklund, M. Fishel and M. Saers. 2008. Mixing and Blending Syntactic and Semantic Dependencies. In *Proc. of CoNLL-2008 Shared Task*.
- W. Sun, H. Li and Z. Sui. 2008. The Integration of Dependency Relation Classification and Semantic Role Labeling Using Bilayer Maximum Entropy Markov Models. In *Proc. of CoNLL-2008 Shared Task*.
- E. F. Tjong Kim San and F. De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proc. of CoNLL-2003*.
- D. Vickrey and D. Koller. 2008. Applying Sentence Simplification to the CoNLL-2008 Shared Task. In *Proc. of CoNLL-2008 Shared Task*.
- R. Weischedel and A. Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical report, Linguistic Data Consortium.
- H. Yamada and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proc. of IWPT*.
- Y. Zhang, R. Wang and H. Uszkoreit. 2008. Hybrid Learning of Dependency Structures from Heterogeneous Linguistic Resources. In *Proc. of CoNLL-2008 Shared Task*.
- Zhao, S. 2005. *Information Extraction from Multiple Syntactic Sources*. Ph.D. thesis, NYU.

A Latent Variable Model of Synchronous Parsing for Syntactic and Semantic Dependencies

James Henderson
Dept Computer Science
Univ Geneva
james.henderson@
cui.unige.ch

Paola Merlo
Dept Linguistics
Univ Geneva
merlo@
lettres.unige.ch

Gabriele Musillo
Depts Linguistics
and Computer Science
Univ Geneva
musillo@
lettres.unige.ch

Ivan Titov*
Dept Computer Science
Univ Illinois at U-C
titov@uiuc.edu

Abstract

We propose a solution to the challenge of the CoNLL 2008 shared task that uses a generative history-based latent variable model to predict the most likely derivation of a synchronous dependency parser for both syntactic and semantic dependencies. The submitted model yields 79.1% macro-average F1 performance, for the joint task, 86.9% syntactic dependencies LAS and 71.0% semantic dependencies F1. A larger model trained after the deadline achieves 80.5% macro-average F1, 87.6% syntactic dependencies LAS, and 73.1% semantic dependencies F1.

1 Introduction

Successes in syntactic tasks, such as statistical parsing and tagging, have recently paved the way to statistical learning techniques for levels of semantic representation, such as recovering the logical form of a sentence for information extraction and question-answering applications (e.g. (Wong and Mooney, 2007)) or jointly learning the syntactic structure of the sentence and the propositional argument-structure of its main predicates (Musillo and Merlo, 2006; Merlo and Musillo, 2008). In this vein, the CoNLL 2008 shared task sets the challenge of learning jointly both syntactic dependencies (extracted from the Penn Treebank (Marcus et al., 1993)) and semantic dependencies (extracted both from PropBank (Palmer et al., 2005)

and NomBank (Meyers et al., 2004) under a unified representation.

We propose a solution that uses a generative history-based model to predict the most likely derivation of a synchronous dependency parser for both syntactic and semantic dependencies. Our probabilistic model is based on Incremental Sigmoid Belief Networks (ISBNs), a recently proposed latent variable model for syntactic structure prediction, which has shown very good behaviour for both constituency (Titov and Henderson, 2007a) and dependency parsing (Titov and Henderson, 2007b). The ability of ISBNs to induce their features automatically enables us to extend this architecture to learning a synchronous parse of syntax and semantics without modification of the main architecture. By solving the problem with synchronous parsing, a probabilistic model is learnt which maximises the joint probability of the syntactic and semantic dependencies and thereby guarantees that the output structure is globally coherent, while at the same time building the two structures separately. This extension of the ISBN architecture is therefore applicable to other problems where two independent, but related, levels of representation are being learnt, such as statistical machine translation.

Currently the largest model we have trained achieves 80.5% macro-average F1 performance for the joint task, 87.6% syntactic dependencies LAS, and 73.1% semantic dependencies F1.

2 The Probability Model

Our probability model is a joint generative model of syntactic and semantic dependencies. The two dependency structures are specified as the sequence of actions for a synchronous parser, which requires each dependency structure to be projec-

©2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

⁰Authors in alphabetical order.

tivised separately.

2.1 Synchronous derivations

The derivations for syntactic dependency trees are the same as specified in (Titov and Henderson, 2007b), which are based on the shift-reduce style parser of (Nivre et al., 2006). The derivations use a stack and an input queue. There are actions for creating a leftward or rightward arc between the top of the stack and the front of the queue, for popping a word from the stack, and for shifting a word from the queue to the stack. The derivations for semantic dependency graphs use virtually the same set of actions, but impose fewer constraints on when they can be applied, due to the fact that a word in a semantic dependency graph can have more than one parent. An additional action $predicate_s$ was introduced to label a predicate with sense s .

Let T_d be a syntactic dependency tree with derivation $D_d^1, \dots, D_d^{m_d}$, and T_s be a semantic dependency graph with derivation $D_s^1, \dots, D_s^{m_s}$. To define derivations for the joint structure T_d, T_s , we need to specify how the two derivations are synchronised, and in particular make the important choice of the granularity of the synchronisation step. Linguistic intuition would perhaps suggest that syntax and semantics are connected at the clause level – a big step size – while a fully integrated system would synchronise at each parsing decision, thereby providing the most communication between these two levels. We choose to synchronise the construction of the two structures at every word – an intermediate step size. This choice is simpler, as it is based on the natural total order of the input, and it avoids the problems of the more linguistically motivated choice, where chunks corresponding to different semantic propositions would be overlapping.

We divide the two derivations into the chunks between shifting each word onto the stack, $c_d^t = D_d^{b_d^t}, \dots, D_d^{e_d^t}$ and $c_s^t = D_s^{b_s^t}, \dots, D_s^{e_s^t}$, where $D_d^{b_d^t-1} = D_s^{b_s^t-1} = shift_{t-1}$ and $D_d^{e_d^t+1} = D_s^{e_s^t+1} = shift_t$. Then the actions of the synchronous derivations consist of quadruples $C^t = (c_d^t, switch, c_s^t, shift_t)$, where $switch$ means switching from syntactic to semantic mode. This gives us the following joint probability model, where n is the number of words in the input.

$$\begin{aligned} P(T_d, T_s) &= P(C^1, \dots, C^n) \\ &= \prod_t P(C^t | C^1, \dots, C^{t-1}) \end{aligned} \quad (1)$$

The probability of each synchronous derivation chunk C^t is the product of four factors, related to the syntactic level, the semantic level and the two synchronising steps.

$$\begin{aligned} P(C^t | C^1, \dots, C^{t-1}) &= \\ &P(c_d^t | C^1, \dots, C^{t-1}) \times \\ &P(switch | c_d^t, C^1, \dots, C^{t-1}) \times \\ &P(c_s^t | switch, c_d^t, C^1, \dots, C^{t-1}) \times \\ &P(shift_t | c_d^t, c_s^t, C^1, \dots, C^{t-1}) \end{aligned} \quad (2)$$

These synchronous derivations C^1, \dots, C^n only require a single input queue, since the $shift$ operations are synchronised, but they require two separate stacks, one for the syntactic derivation and one for the semantic derivation.

The probability of c_d^t is decomposed into derivation action D^i probabilities, and likewise for c_s^t .

$$\begin{aligned} P(c_d^t | C^1, \dots, C^{t-1}) &= \\ &= \prod_i P(D_d^i | D_d^{b_d^i}, \dots, D_d^{i-1}, C^1, \dots, C^{t-1}) \end{aligned} \quad (3)$$

The actions are also sometimes split into a sequence of elementary decisions $D^i = d_1^i, \dots, d_n^i$, as discussed in (Titov and Henderson, 2007a).

2.2 Projectivisation of dependencies

These derivations can only specify projective syntactic or semantic dependency graphs. Exploratory data analysis indicates that many instances of non-projectivity in the complete graph are due to crossings of the syntactic and semantic graphs. The amount of non-projectivity of the joint syntactic-semantic graph is approximately 7.5% non-projective arcs, while summing the non-projectivity within the two separate graphs results in only roughly 3% non-projective arcs.

Because our synchronous derivations use two different stacks for the syntactic and semantic dependencies, respectively, we only require each individual graph to be projective. As with many dependency parsers (Nivre et al., 2006; Titov and Henderson, 2007b), we handle non-projective (i.e. crossing) arcs by transforming them into non-crossing arcs with augmented labels.¹ Because our syntactic derivations are equivalent to those of (Nivre et al., 2006), we use their HEAD methods to projectivise the syntactic dependencies.

Although our semantic derivations use the same set of actions as the syntactic derivations, they differ in that the graph of semantic dependencies need

¹During testing, these projectivised structures are then transformed back to the original format for evaluation.

not form a tree. The only constraints we place on the set of semantic dependencies are imposed by the use of a stack, which excludes crossing arcs. Given two crossing arcs, we try to uncross them by changing an endpoint of one of the arcs. The arc (p, a) , where p is a predicate and a is an argument, is changed to (p, h) , where h is the syntactic head of argument a . Its label r is then changed to r/d where d is the syntactic dependency of a on h . This transformation may need to be repeated before the arcs become uncrossed. The choice of which arc to transform is done using a greedy algorithm and a number of heuristics, without doing any global optimisation across the data.

This projectivisation method is similar to the HEAD method of (Nivre et al., 2006), but has two interesting new characteristics. First, syntactic dependencies are used to projectivise the semantic dependencies. Because the graph of semantic roles is disconnected, moving across semantic arcs is often not possible. This would cause a large number of roles to be moved to ROOT. Second, our method changes the semantic argument of a given predicate, whereas syntactic dependency projectivisation changes the head of a given dependent. This difference is motivated by a predicate-centred view of semantic dependencies, as it avoids changing a predicate to a node which is not a predicate.

3 The Learning Architecture

The synchronous derivations described above are modelled with an Incremental Sigmoid Belief Network (ISBN) (Titov and Henderson, 2007a). ISBNs are dynamic Bayesian Networks which incrementally specify their model structure based on the partial structure being built by a derivation. They have previously been applied to constituency and dependency parsing. In both cases the derivations were based on a push-down automaton, but ISBNs can be directly applied to any automaton. We successfully apply ISBNs to a two-stack automaton, without changing the machine learning methods.

3.1 The Incremental Sigmoid Belief Networks

ISBNs use vectors of latent variables to represent properties of parsing history relevant to the next decisions. Latent variables do not need to be annotated in the training data, but instead get induced during learning. As illustrated by the vectors S^i in figure 1, the latent feature vectors are used to estimate the probabilities of derivation actions D^i .

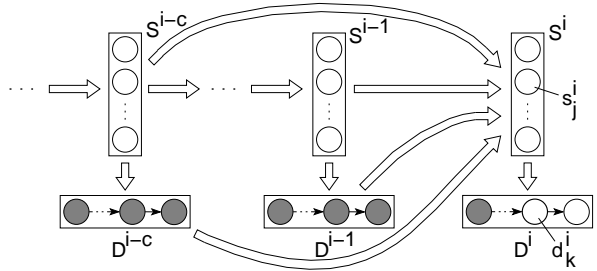


Figure 1: An ISBN for estimating $P(d_k^i | history(i, k))$ – one of the elementary decisions. Variables whose values are given in $history(i, k)$ are shaded, and latent and current decision variables are unshaded.

Latent variable vectors are connected to variables from previous positions via a pattern of edges determined by the previous decisions. Our ISBN model distinguishes two types of latent states: syntactic states, when syntactic decisions are considered, and semantic states, when semantic decision are made. Different patterns of interconnections are used for different types of states. We use the neural network approximation (Titov and Henderson, 2007a) to perform inference in our model.

As also illustrated in figure 1, the induced latent variables S^i at state i are statistically dependent on both pre-defined features of the derivation history D^1, \dots, D^{i-1} and the latent variables for a finite set of relevant previous states $S^{i'}, i' < i$. Choosing this set of relevant previous states is one of the main design decisions in building an ISBN model. By connecting to a previous state, we place that state in the local context of the current decision. This specification of the domain of locality determines the inductive bias of learning with ISBNs. Thus, we need to choose the set of local (i.e. connected) states in accordance with our prior knowledge about which previous decisions are likely to be particularly relevant to the current decision.

3.2 Layers and features

To choose previous relevant decisions, we make use of the partial syntactic and semantic dependency structures which have been decided so far in the parse. Specifically, the current latent state vector is connected to the most recent previous latent state vectors (if they exist) whose configuration shares a node with the current configuration, as specified in Table 1. The nodes are chosen because their properties are thought to be relevant to the current decision. Each row of the table indicates which nodes need to be identical, while each

Closest	Current	Syn-Syn	Srl-Srl	Syn-Srl
Input	Input	+	+	+
Top	Top	+	+	+
RDT	Top	+	+	
LDT	Top	+	+	
HT	Top	+	+	
LDN	Top	+	+	
Input	Top	+		

Table 1: Latent-to-latent variable connections. Input= input queue; Top= top of stack; RDT= rightmost right dependent of top; LDT= leftmost left dependent of top; HT= Head of top; LDN= leftmost dependent of next (front of input).

column indicates whether the latent state vectors are for the syntactic or semantic derivations. For example, the first row indicates edges between the current state and a state which had the same input as the current state. The three columns indicate that this edge holds within syntactic states, within semantic states, and from syntactic to semantic states. The fourth cell of the third row, for example, indicates that there is an edge between the current semantic state on top of the stack and the most recent semantic state where the rightmost dependent of the current top of the semantic stack was at the top of the semantic stack.

Each of these relations has a distinct weight matrix for the resulting edges in the ISBN, but the same weight matrix is used at each position where the relation applies. Training and testing times scale linearly with the number of relations.

The pre-defined features of the parse history which also influence the current decision are specified in table 2. The model distinguishes argument roles of nominal predicates from argument roles of verbal predicates.

3.3 Decoding

Given a trained ISBN as our probability estimator, we search for the most probable joint syntactic-semantic dependency structure using a beam search. Most pruning is done just after each *shift* operation (when the next word is predicted). Global constraints (such as label uniqueness) are not enforced by decoding, but can be learnt.

For the system whose results we submitted, we then do a second step to improve on the choice of syntactic dependency structure. Because of the lack of edges in the graphical model from semantic to syntactic states, it is easy to marginalise out the semantic structure, giving us the most probable syntactic dependency structure. This syntactic structure is then combined with the semantic struc-

State	Stack	Syntactic step features			
		LEX	POS	DEP	
Input		+	+		
Top	syn	+	+		
Top - 1	syn		+		
HT	syn	+			
RDT	syn			+	
LDT	syn			+	
LDN	syn			+	
State	Stack	Semantic step features			
		LEX	POS	DEP	SENSE
Input		+	+		+
Top	sem	+	+		+
Top - 1	sem	+	+		
HT	sem			+	
RDT	sem			+	
LDT	sem			+	
LDN	sem			+	
A0-A5 of Top	sem		+		
A0-A5 of Input	sem		+		

Table 2: Pre-defined features. syn=syntactic stack; sem=semantic stack. Input= input queue; Top= top of stack; RDT= rightmost dependent of top; LDT= leftmost dependent of Top; HT= Head of top; LDN= leftmost dependent of next (front of input); A0-A5 of Top/Input= arguments of top of stack / input.

ture from the first stage, to get our submitted results. This second stage does not maximise performance on the joint syntactic-semantic dependency structure, but it better fits the evaluation measure used to rank systems.

4 Experiments and Discussion

The experimental set-up common for all the teams is described in the introduction (Surdeanu et al., 2008). The submitted model has latent variable vectors of 60 units, and a word frequency cut-off of 100, resulting in a small vocabulary of 1083 words. We used a beam of size 15 to prune derivations after each *shift* operation to obtain the joint structure, and a beam of size 40 when performing the marginalisation. Training took approximately 2.5 days on a standard PC with 3.0 GHz Pentium4 CPU. It took approximately 2 hours to parse the entire testing set (2,824 sentences) and an additional 3 hours to perform syntactic parsing when marginalising out the semantic structures.² Shortly after the submission deadline, we trained a ‘large’ model with a latent variable vector of size 80, a word frequency cut-off of 20, and additional latent-to-latent connections from semantics to syntax of the same configuration as the last column

²A multifold speed-up with a small decrease in accuracy can be achieved by using a small beam.

	Syn	Semantic			Overall		
	LAS	P	R	F1	P	R	F1
Submitted							
D	86.1	78.8	64.7	71.1	82.5	75.4	78.8
W	87.8	79.6	66.2	72.3	83.7	77.0	80.2
B	80.0	66.6	55.3	60.4	73.3	67.6	70.3
WB	86.9	78.2	65.0	71.0	82.5	76.0	79.1
Joint inference							
D	85.5	78.8	64.7	71.1	82.2	75.1	78.5
Large, joint inference							
D	86.5	79.9	67.5	73.2	83.2	77.0	80.0
W	88.5	80.4	69.2	74.4	84.4	78.8	81.5
B	81.0	68.3	57.7	62.6	74.7	69.4	71.9
WB	87.6	79.1	67.9	73.1	83.4	77.8	80.5

Table 3: Scores on the development set and the final testing sets (percentages). D= development set; W=WSJ; B=Brown; WB=WSJ+Brown;

of table 1. This model took about 50% longer in training and testing.

In table 3, we report results for the marginalised inference (‘submitted’) and joint inference for the submitted model, and the results for joint inference with the ‘large’ model. The larger model improves on the submitted results by almost 1.5%, a significant improvement. If completed earlier, this model would have been fifth overall, second for syntactic LAS, and fifth for semantic F1.

To explore the relationship between the two components of the model, we removed the edges between the syntax and the semantics in the submitted model. This model’s performance drops by about 3.5% for semantic role labelling, thereby indicating that the latent annotation of parsing states helps semantic role labelling. However, it also indicates that there is much room for improvement in developing useful semantic-specific features, which was not done for these experiments simply due to constraints on development time.

To test whether joint learning degrades the accuracy of the syntactic parsing model, we trained a syntactic parsing model with the same features and the same pattern of interconnections as used for the syntactic states in our joint model. The resulting labelled attachment score was non-significantly lower (0.2%) than the score for the marginalised inference with the joint model. This result suggests that, though the latent variables associated with syntactic states in the joint model were trained to be useful in semantic role labelling, this did not have a negative effect on syntactic parsing accuracy, and may even have helped.

Finally, an analysis of the errors on the development set for the submitted model paints a coherent picture. We find attachment of adjuncts particu-

larly hard. For dependency labels, we make the most mistakes on modification labels, while for semantic labels, we find TMP, ADV, LOC, and PRN particularly hard. NomBank arcs are not learnt as well as PropBank arcs: we identify PropBank SRL arguments at F1 70.8% while Nombank arguments reach 58.1%, and predicates at accuracy 87.9% for PropBank and 74.9% for NomBank.

5 Conclusions

While still preliminary, these results indicate that synchronous parsing is an effective way of building joint models on separate structures. The generality of the ISBN design used so far suggests that ISBN’s latent feature induction extends well to estimating very complex probability models, with little need for feature engineering. Nonetheless, performance could be improved by task-specific features, which we plan for future work.

Acknowledgements

This work was partly funded by European Community FP7 grant 216594 (CLASSiC, www.classic-project.org), Swiss NSF grant 114044, and Swiss NSF fellowships PBGE2-117146 and PBGE22-119276. Part of this work was done when G. Musillo was visiting MIT/CSAIL, hosted by Prof. Michael Collins.

References

- Marcus, M., B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.
- Merlo, P. and G. Musillo. 2008. Semantic parsing for high-precision semantic role labelling. In *Procs of CoNLL 2008*, Manchester, UK.
- Meyers, A., R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The nombank project: An interim report. In Meyers, A., editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, 24–31, Boston, MA.
- Musillo, G. and P. Merlo. 2006. Accurate semantic parsing of the Proposition Bank. In *Procs of NAACL 2006*, New York, NY.
- Nivre, J., J. Hall, J. Nilsson, G. Eryigit, and S. Marinov. 2006. Pseudo-projective dependency parsing with support vector machines. In *Proc. of CoNLL*, 221–225, New York, USA.
- Palmer, M., D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31:71–105.
- Surdeanu, M., R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Procs of CoNLL-2008*, Manchester, UK.
- Titov, I. and J. Henderson. 2007a. Constituent parsing with incremental sigmoid belief networks. In *Procs of ACL’07*, pages 632–639, Prague, Czech Republic.
- Titov, I. and J. Henderson. 2007b. A latent variable model for generative dependency parsing. In *Procs of IWPT’07*, Prague, Czech Republic.
- Wong, Y.W. and R. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Procs of ACL’07*, 960–967, Prague, Czech Republic.

Dependency-based Syntactic–Semantic Analysis with PropBank and NomBank

Richard Johansson and Pierre Nugues

Lund University, Sweden

{richard, pierre}@cs.lth.se

Abstract

This paper presents our contribution in the closed track of the 2008 CoNLL Shared Task (Surdeanu et al., 2008). To tackle the problem of joint syntactic–semantic analysis, the system relies on a syntactic and a semantic subcomponent. The syntactic model is a bottom-up projective parser using pseudo-projective transformations, and the semantic model uses global inference mechanisms on top of a pipeline of classifiers. The complete syntactic–semantic output is selected from a candidate pool generated by the subsystems.

The system achieved the top score in the closed challenge: a labeled syntactic accuracy of 89.32%, a labeled semantic F1 of 81.65, and a labeled macro F1 of 85.49.

1 Introduction: Syntactic–Semantic Analysis

Intuitively, semantic interpretation should help syntactic disambiguation, and joint syntactic–semantic analysis has a long tradition in linguistic theory. This motivates a statistical modeling of the problem of finding a syntactic tree \hat{y}_{syn} and a semantic graph \hat{y}_{sem} for a sentence x as maximizing a function F that scores the joint syntactic–semantic structure:

$$\langle \hat{y}_{syn}, \hat{y}_{sem} \rangle = \arg \max_{y_{syn}, y_{sem}} F(x, y_{syn}, y_{sem})$$

The dependencies in the feature representation used to compute F determine the tractability of the search procedure needed to perform the maximization. To be able to use complex syntactic features

such as paths when predicting semantic structures, exact search is clearly intractable. This is true even with simpler feature representations – the problem is a special case of multi-headed dependency analysis, which is NP-hard even if the number of heads is bounded (Chickering et al., 1994).

This means that we must resort to a simplification such as an incremental method or a reranking approach. We chose the latter option and thus created syntactic and semantic submodels. The joint syntactic–semantic prediction is selected from a small list of candidates generated by the respective subsystems.

2 Syntactic Submodel

We model the process of syntactic parsing of a sentence x as finding the parse tree $\hat{y}_{syn} = \arg \max_y F(x, y)$ that maximizes a scoring function F . The learning problem consists of fitting this function so that the cost of the predictions is as low as possible according to a cost function ρ . In this work, we consider linear scoring functions of the following form:

$$F(x, y) = w \cdot \Psi(x, y)$$

where $\Psi(x, y)$ is a numeric feature representation of the pair (x, y) and w a vector of feature weights. We defined the syntactic cost ρ as the sum of link costs, where the link cost was 0 for a correct dependency link with a correct label, 0.5 for a correct link with an incorrect label, and 1 for an incorrect link.

A widely used framework for fitting the weight vector is the max-margin model (Taskar et al., 2003), which is a generalization of the well-known support vector machines to general cost-based prediction problems. Since the large number of training examples and features in our case make an exact solution of the max-margin optimization problem impractical, we used the online passive–aggressive algorithm (Crammer et al.,

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

2006), which approximates the optimization process in two ways:

- The weight vector w is updated incrementally, one example at a time.
- For each example, only the most violated constraint is considered.

The algorithm is a margin-based variant of the perceptron (preliminary experiments show that it outperforms the ordinary perceptron on this task). Algorithm 1 shows pseudocode for the algorithm.

Algorithm 1 The Online PA Algorithm

```

input Training set  $\mathcal{T} = \{(\mathbf{x}_t, y_t)\}_{t=1}^T$ 
      Number of iterations  $N$ 
      Regularization parameter  $C$ 
Initialize  $w$  to zeros
repeat  $N$  times
  for  $(\mathbf{x}_t, y_t)$  in  $\mathcal{T}$ 
    let  $\tilde{y}_t = \arg \max_y F(\mathbf{x}_t, y) + \rho(y_t, y)$ 
    let  $\tau_t = \min \left( C, \frac{F(\mathbf{x}_t, \tilde{y}_t) - F(\mathbf{x}_t, y_t) + \rho(y_t, \tilde{y}_t)}{\|\Psi(\mathbf{x}, y_t) - \Psi(\mathbf{x}, \tilde{y}_t)\|^2} \right)$ 
     $w \leftarrow w + \tau_t (\Psi(\mathbf{x}, y_t) - \Psi(\mathbf{x}, \tilde{y}_t))$ 
return  $w_{\text{average}}$ 

```

We used a C value of 0.01, and the number of iterations was 6.

2.1 Features and Search

The feature function Ψ is a second-order edge-factored representation (McDonald and Pereira, 2006; Carreras, 2007). The second-order representation allows us to express features not only of head-dependent links, but also of siblings and children of the dependent. This feature set forces us to adopt the expensive search procedure by Carreras (2007), which extends Eisner’s span-based dynamic programming algorithm (1996) to allow second-order feature dependencies. Since the cost function ρ is based on the cost of single links, this procedure can also be used to find the maximizer of $F(\mathbf{x}_i, y_{ij}) + \rho(y_i, y_{ij})$, which is needed at training time. The search was constrained to disallow multiple root links.

2.2 Handling Nonprojective Links

Although only 0.4% of the links in the training set are nonprojective, 7.6% of the sentences contain at least one nonprojective link. Many of these links represent long-range dependencies – such as *wh*-movement – that are valuable for semantic processing. Nonprojectivity cannot be handled by span-based dynamic programming algorithms. For

parsers that consider features of single links only, the Chu-Liu/Edmonds algorithm can be used instead. However, this algorithm cannot be generalized to the second-order setting – McDonald and Pereira (2006) proved that this problem is NP-hard, and described an approximate greedy search algorithm.

To simplify implementation, we instead opted for the pseudo-projective approach (Nivre and Nilsson, 2005), in which nonprojective links are lifted upwards in the tree to achieve projectivity, and special trace labels are used to enable recovery of the nonprojective links at parse time. The use of trace labels in the pseudo-projective transformation leads to a proliferation of edge label types: from 69 to 234 in the training set, many of which occur only once. Since the running time of our parser depends on the number of labels, we used only the 20 most frequent trace labels.

3 Semantic Submodel

Our semantic model consists of three parts:

- A SRL classifier pipeline that generates a list of candidate predicate–argument structures.
- A constraint system that filters the candidate list to enforce linguistic restrictions on the global configuration of arguments.
- A global classifier that rescores the predicate–argument structures in the filtered candidate list.

Rather than training the models on gold-standard syntactic input, we created an automatically parsed training set by 5-fold cross-validation. Training on automatic syntax makes the semantic classifiers more resilient to parsing errors, in particular adjunct labeling errors.

3.1 SRL Pipeline

The SRL pipeline consists of classifiers for predicate identification, predicate disambiguation, support identification (for noun predicates), argument identification, and argument classification. We trained one set of classifiers for verb predicates and another for noun predicates. For the predicate disambiguation classifiers, we trained one subclassifier for each lemma. All classifiers in the pipeline were L2-regularized linear logistic regression classifiers, implemented using the efficient LIBLINEAR package (Lin et al., 2008). For multi-class problems, we used the one-vs-all binarization

method, which makes it easy to prevent outputs not allowed by the PropBank or NomBank frame.

Since our classifiers were logistic, their output values could be meaningfully interpreted as probabilities. This allowed us to combine the scores from subclassifiers into a score for the complete predicate–argument structure. To generate the candidate lists used by the global SRL models, we applied beam search based on these scores using a beam width of 4.

The features used by the classifiers are listed in Tables 1 and 2. In the tables, the features used by the classifiers for noun and verb predicates are indicated by N and V, respectively. We selected the feature sets by greedy forward subset selection.

Feature	PredId	PredDis
PREDWORD	N,V	N,V
PREDLEMMA	N,V	N,V
PREDPARENTWORD/POS	N,V	N,V
CHILDEPSET	N,V	N,V
CHILDWORDSET	N,V	N,V
CHILDWORDDEPSET	N,V	N,V
CHILDPOSSET	N,V	N,V
CHILDPOSDEPSET	N,V	N,V
DEPSUBCAT	N,V	N,V
PREDRELTOPARENT	N,V	N,V

Table 1: Classifier features in predicate identification and disambiguation.

Feature	Supp	ArgId	ArgCl
PREDPARENTWORD/POS	N	N,V	
CHILDEPSET	N	N,V	N,V
PREDLEMMASENSE	N	N,V	N,V
VOICE		V	V
POSITION	N	N,V	N,V
ARGWORD/POS	N	N,V	N,V
LEFTWORD/POS		N	N,V
RIGHTWORD/POS	N	N,V	N,V
LEFTSIBLINGWORD/POS			N,V
RIGHTSIBLINGWORD/POS		N	N
PREDPOS	N	N,V	V
RELPATH	N	N,V	N,V
POSPATH		N	
RELPATHTOSUPPORT		N	N
VERBCHAINHASSUBJ		V	V
CONTROLLERHASOBJ		V	N
PREDRELTOPARENT	N	N,V	N,V
FUNCTION			N,V

Table 2: Classifier features in argument identification and classification and support detection.

Features Used in Predicate Identification and Disambiguation

PREDWORD, PREDLEMMA. The lexical form and lemma of the predicate.

PREDPARENTWORD and PREDPARENTPOS. Form and part-of-speech tag of the parent node of the predicate.

CHILDEPSET, CHILDWORDSET, CHILDWORDDEPSET, CHILDPOSSET, CHILDPOSDEPSET. These features represent the set of dependents of the predicate using combinations of dependency labels, words, and parts of speech.

DEPSUBCAT. Subcategorization frame: the concatenation of the dependency labels of the predicate dependents.

PREDRELTOPARENT. Dependency relation between the predicate and its parent.

Features Used in Argument Identification and Classification

PREDLEMMASENSE. The lemma and sense number of the predicate, e.g. *give.01*.

VOICE. For verbs, this feature is Active or Passive. For nouns, it is not defined.

POSITION. Position of the argument with respect to the predicate: Before, After, or On.

ARGWORD and ARGPOS. Lexical form and part-of-speech tag of the argument node.

LEFTWORD, LEFTPOS, RIGHTWORD, RIGHTPOS. Form/part-of-speech tag of the leftmost/rightmost dependent of the argument.

LEFTSIBLINGWORD, LEFTSIBLINGPOS, RIGHTSIBLINGWORD, RIGHTSIBLINGPOS. Form/part-of-speech tag of the left/right sibling of the argument.

PREDPOS. Part-of-speech tag of the predicate.

RELPATH. A representation of the complex grammatical relation between the predicate and the argument. It consists of the sequence of dependency relation labels and link directions in the path between predicate and argument, e.g. $IM\uparrow OPRD\uparrow OBJ\downarrow$.

POSPATH. An alternative view of the grammatical relation, which consists of the POS tags passed when moving from predicate to argument, e.g. $VB\uparrow TO\uparrow VBP\downarrow PRP$.

RELPATHTOSUPPORT. The RELPATH from the argument to a support chain.

VERBCHAINHASSUBJ. Binary feature that is set to true if the predicate verb chain has a subject. The purpose of this feature is to resolve verb coordination ambiguity as in Figure 1.

CONTROLLERHASOBJ. Binary feature that is true if the link between the predicate verb chain and its parent is OPRD, and the parent has an object. This feature is meant to resolve control ambiguity as in Figure 2.

FUNCTION. The grammatical function of the argument node. For direct dependents of the predicate, this is identical to the RELPATH.

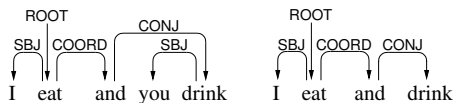


Figure 1: Coordination ambiguity: The subject *I* is in an ambiguous position with respect to *drink*.

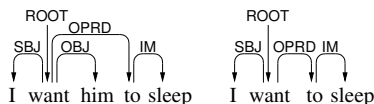


Figure 2: Subject/object control ambiguity: *I* is in an ambiguous position with respect to *sleep*.

3.2 Linguistically Motivated Global Constraints

The following three global constraints were used to filter the candidates generated by the pipeline.

CORE ARGUMENT CONSISTENCY. Core argument labels must not appear more than once.

DISCONTINUITY CONSISTENCY. If there is a label C-X, it must be preceded by a label X.

REFERENCE CONSISTENCY. If there is a label R-X and the label is inside a relative clause, it must be preceded by a label X.

3.3 Global SRL Model

Toutanova et al. (2005) have showed that a global model that scores the complete predicate–argument structure can lead to substantial performance gains. We therefore created a global SRL classifier using the following global features in addition to the features from the pipeline:

CORE ARGUMENT LABEL SEQUENCE. The complete sequence of core argument labels. The sequence also includes the predicate and voice, for instance *A0+break.01/Active+A1*.

MISSING CORE ARGUMENT LABELS. The set of core argument labels declared in the PropBank/NomBank frame that are not present in the predicate–argument structure.

Similarly to the syntactic submodel, we trained the global SRL model using the online passive–aggressive algorithm. The cost function ρ was

defined as the number of incorrect links in the predicate–argument structure. The number of iterations was 20 and the regularization parameter C was 0.01. Interestingly, we noted that the global SRL model outperformed the pipeline even when no global features were added. This shows that the global learning model can correct label bias problems introduced by the pipeline architecture.

4 Syntactic–Semantic Integration

Our baseline joint feature representation contained only three features: the log probability of the syntactic tree and the log probability of the semantic structure according to the pipeline and the global model, respectively. This model was trained on the complete training set using cross-validation. The probabilities were obtained using the multinomial logistic function (“softmax”).

We carried out an initial experiment with a more complex joint feature representation, but failed to improve over the baseline. Time prevented us from exploring this direction conclusively.

5 Results

The submitted results on the development and test corpora are presented in the upper part of Table 3. After the submission deadline, we corrected a bug in the predicate identification method. This resulted in improved results shown in the lower part.

Corpus	Syn acc	Sem F1	Macro F1
Development	88.47	80.80	84.66
Test WSJ	90.13	81.75	85.95
Test Brown	82.81	69.06	75.95
Test WSJ + Brown	89.32	80.37	84.86
Development	88.47	81.86	85.17
Test WSJ	90.13	83.75	86.61
Test Brown	82.84	69.85	76.34
Test WSJ + Brown	89.32	81.65	85.49

Table 3: Results.

5.1 Syntactic Results

Table 4 shows the effect of adding second-order features to the parser in terms of accuracy as well as training and parsing time on a Mac Pro, 3.2 GHz. The training times were measured on the complete training set and the parsing time and accuracies on the development set. Similarly to Carreras (2007), we see that these features have a very large impact on parsing accuracy, but also that the parser pays dearly in terms of efficiency as the search complexity increases from $O(n^3)$ to $O(n^4)$.

Since the low efficiency of the second-order parser restricts its use to batch applications, we see an interesting research direction to find suitable compromises between the two approaches, for instance by sacrificing the exact search procedure.

System	Training	Parse	Labeled	Unlabeled
1st order	65 min	28 sec	85.78	89.51
2nd order	60 hours	34 min	88.33	91.43

Table 4: Impact of second-order features.

Table 5 shows the dependency types most affected by the addition of second-order features to the parser when ordered by the increase in F1. As can be seen, they are all verb adjunct categories, which demonstrates the effect of grandchild features on PP attachment and labeling.

Label	ΔR	ΔP	ΔF_1
TMP	14.7	12.9	13.9
DTV	0	19.9	10.5
LOC	7.8	12.3	9.9
PRP	12.4	6.7	9.6
DIR	5.9	7.2	6.5

Table 5: Labels affected by second-order features.

5.2 Semantic Results

To assess the effect of the components in the semantic submodel, we tested their performance on the top-scoring parses from the syntactic model. Table 6 shows the results. The baseline system consists of the SRL pipeline only (P). Adding linguistic constraints (C) results in a more precision-oriented system with slightly lower recall, but significantly higher F1. Even higher performance is obtained when adding the global SRL model (G).

System	P	R	F1
P	80.74	77.98	79.33
P+C	82.42	77.66	79.97
P+C+G	83.64	78.14	80.40

Table 6: SRL results on the top-scoring parse trees.

5.3 Syntactic–Semantic Integration

The final experiment concerned the integration of syntactic and semantic analysis. In this setting, the system chooses the output that maximizes the joint syntactic–semantic score, based on the top N syntactic trees. Table 7 shows the results on the development set. We see that syntactic–semantic integration improves both syntactic accuracy and

semantic F1. This holds for the constraint-based SRL system as well as for the full system.

Sem model	N	Syn acc	Sem F1	Macro F1
P+C	1	88.33	79.97	84.17
P+C	16	88.42	80.42	84.44
P+C+G	1	88.33	80.40	84.39
P+C+G	16	88.47	80.80	84.66

Table 7: Syntactic–semantic integration.

6 Conclusion

We have described a system¹ for syntactic and semantic dependency analysis based on PropBank and NomBank, and detailed the implementation of its subsystems. Crucial to our success was the high performance of the syntactic parser, which achieved a high accuracy. In addition, we reconfirmed the benefits of global inference in semantic analysis: both constraint-based and learning-based methods resulted in improvements over a baseline. Finally, we showed that integration of syntactic and semantic analysis is beneficial for both sub-tasks. We hope that this shared task will spur further research that leads to new feature representations and search procedures to handle the problem of joint syntactic and semantic analysis.

References

- Carreras, Xavier. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of CoNLL*.
- Chickering, David M., Dan Geiger, and David Heckerman. 1994. Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Microsoft Research.
- Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 2006(7):551–585.
- Eisner, Jason M. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of ICCL*.
- Lin, Chih-Jen, Ruby C. Weng, and S. Sathya Keerthi. 2008. Trust region Newton method for large-scale logistic regression. *JMLR*, 2008(9):627–650.
- McDonald, Ryan and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL-2006*.
- Nivre, Joakim and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL-2005*.
- Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL–2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL–2008*.
- Taskar, Ben, Carlos Guestrin, and Daphne Koller. 2003. Max-margin Markov networks. In *Proceedings of NIPS*.
- Toutanova, Kristina, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL-2005*.

¹Our system is freely available for download at http://nlp.cs.lth.se/lth_srl.

A Joint Model for Parsing Syntactic and Semantic Dependencies

Xavier Lluís and Lluís Màrquez

TALP Research Centre – Software Department (LSI)

Technical University of Catalonia (UPC)

{xlluis, lluis}@lsi.upc.edu

Abstract

This paper describes a system that jointly parses syntactic and semantic dependencies, presented at the CoNLL-2008 shared task (Surdeanu et al., 2008). It combines online Perceptron learning (Collins, 2002) with a parsing model based on the Eisner algorithm (Eisner, 1996), extended so as to jointly assign syntactic and semantic labels. Overall results are 78.11 global F_1 , 85.84 LAS, 70.35 semantic F_1 . Official results for the shared task (63.29 global F_1 ; 71.95 LAS; 54.52 semantic F_1) were significantly lower due to bugs present at submission time.

1 Introduction

The main goal of this work was to construct a joint learning architecture for syntactic-semantic parsing and to test whether the syntactic and semantic layers can benefit each other from the global training and inference.

All the components of our system were built from scratch for this shared task. Due to strong time limitations, our design decisions were biased towards constructing a simple and feasible system. Our proposal is a first order linear model that relies on an online averaged Perceptron for learning (Collins, 2002) and an extended Eisner algorithm for the joint parsing inference.

Systems based on Eisner algorithm (Carreras et al., 2006; Carreras, 2007) showed a competitive performance in the syntactic parsing of the English language in some past CoNLL shared tasks. Also,

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

we believe that extending the Eisner algorithm to jointly parse syntactic and semantic dependencies it is a natural step to follow.

Note that syntactic and semantic tasks are related but not identical. Semantic dependencies can take place between words loosely related by the syntactic structure. Another difficulty is that state of the art SRL systems (Surdeanu et al., 2007) strongly rely on features extracted from the syntactic tree. The joint model grows syntactic and semantic structures at the same time, so features extracted from the syntactic tree (e.g., a syntactic path between a modifier and a distant predicate) are not available or expensive to compute within the Eisner algorithm. We overcome this problem again with a very simple (though not elegant) solution, consisting of introducing a previous syntactic parsing step.

2 System architecture

This section briefly describes the main components of our system: 1) Preprocessing, 2) Syntactic parsing, 3) Predicate identification, 4) Joint syntactic-semantic parsing, and 5) Postprocessing.

In *preprocessing*, the training corpus is traversed and feature extraction performed. Main features are borrowed from pre-existing well-known systems (see next subsection). The initial *syntactic parsing* is based on an Eisner parser trained with Perceptron and it is merely intended to allow the extraction of syntactic-based features for all the following phases (which share exactly the same feature set extracted from these parse trees). *Predicate identification* recognizes predicates by applying SVM classifiers¹ and a set of simple heuristic rules. The *joint syntactic-semantic parsing* phase

¹We used SVM-light (see www.joachims.org for details).

is the core module of this work. It simultaneously derives the syntactic and semantic dependencies by using a first order Eisner model, extended with semantic labels and trained with averaged Perceptron. Finally, *postprocessing* simply selects the most frequent sense for each predicate.

2.1 Preprocessing and feature extraction

All features in our system are calculated in the preprocessing phase. We use the features described in McDonald et al. (2005) and Carreras et al. (2006) as input for the syntactic parsing phase, except for the dynamic features from Carreras et al. (2006). The joint syntactic-semantic parser uses all the previous features and also specific features for semantic parsing from Xue and Palmer (2004) and Surdeanu et al. (2007). The features have been straightforwardly adapted to the dependency structure used in this shared task, by substituting any reference to a syntactic constituent by the head of that constituent. About 5M features were extracted from the training corpus. The number of features was reduced to $\sim 222\text{K}$ using a frequency threshold filter. A detailed description of the feature set can be found at Lluís (Forthcoming 2008).

2.2 Syntactic parsing

Our system uses the Eisner algorithm combined with an online averaged Perceptron. We define the basic model, which is also the starting point for the joint model. Let L be the set of syntactic labels, $x = x_1, \dots, x_n$ a sentence with n words, and $\mathcal{Y}(x)$ the set of all possible projective dependency trees for x .

A dependency tree $y \in \mathcal{Y}(x)$ is a labeled tree with arcs of the form $\langle h, m, l \rangle$ that is rooted on an artificial node, 0, added for this purpose. The head, h , and modifier, m , for a dependency index words in the sentence and can take values in $0 \leq h \leq n$ and $1 \leq m \leq n$. $l \in L$ is the label of the dependency.

The dependency parser (dp) is interested in finding the best scored tree for a given sentence x :

$$\text{dp}(x, \mathbf{w}) = \arg \max_{y \in \mathcal{Y}(x)} \text{score_tree}(y, x, \mathbf{w})$$

Using an arc-based first order factorization, the function $\text{score_tree}(y, x, \mathbf{w})$ is defined as the summation of scores of the dependencies in y :

$$\sum_{\langle h, m, l \rangle \in y} \text{score}(\langle h, m, l \rangle, x, \mathbf{w}),$$

where \mathbf{w} is the weight vector of the parser, computed using an online perceptron. The weight vector \mathbf{w} can be seen as a concatenation of $|L|$ weight vectors of d components, one for each of the labels: $\mathbf{w} = (\mathbf{w}^{(1)}, \dots, \mathbf{w}^{(l)}, \dots, \mathbf{w}^{(|L|)})$. A function ϕ is assumed to extract features from a dependency $\langle h, m, l \rangle$ and from the whole sentence x . This function represents the extracted features as a d -dimensional vector.

With all these elements, the score of a dependency $\langle h, m, l \rangle$ is computed as a linear function:

$$\text{score}(\langle h, m, l \rangle, x, \mathbf{w}) = \phi(\langle h, m, l \rangle, x) \cdot \mathbf{w}^{(l)}$$

2.3 Predicate identification

We identified as verb predicates all verbs excluding the auxiliaries and the verb *to be*. These simple rules based on the POS and lemma of the tokens are enough to correctly identify almost all verb predicates. With regard to noun predicates, we directly identified as predicates the lemmas which appeared always as predicates with a minimum frequency in the training corpus. The remaining noun predicates were identified by a degree-2 polynomial SVM. This classifier was trained with the same features used in subsequent phases, but excluding those requiring identified predicates.

2.4 Joint syntactic and semantic Parsing

The previously described basic parsing model will be extended to jointly assign semantic dependency labels. Let S be the set of semantic labels. Note that at this point, a sentence x has a set of q words already identified as predicates. We will refer to them as p_1, \dots, p_q , where $p_i \in \{1, \dots, n\}$. We consider that each dependency has a set of semantic tags $l_{sem\ p_1}, \dots, l_{sem\ p_q}$ one for each sentence predicate p_i . Also, we consider an extra *no-argument* label in the set of semantic labels S . Thus, an extended dependency d_s is defined as:

$$d_s = \langle h, m, l_{syn}, l_{sem\ p_1}, \dots, l_{sem\ p_q} \rangle,$$

where l_{syn} denotes the syntactic label for the dependency.

Again, the best parse tree is that maximizing the score of a first order factorization:

$$\text{dp}(x, \mathbf{w}, y') = \arg \max_{y \in \mathcal{Y}(x)} \text{score_tree}(y, x, \mathbf{w}, y')$$

$$\begin{aligned} \text{score_tree}(y, x, \mathbf{w}, y') &= \\ &= \sum_{\langle h, m, l \rangle \in y} \text{score}(\langle h, m, l \rangle, x, \mathbf{w}, y'), \end{aligned}$$

where the dependency label is now extended to $\mathbf{l} = \langle l_{syn}, l_{sem\ p_1}, \dots, l_{sem\ p_q} \rangle$ and y' denotes the precomputed syntax tree. The score of a syntactic-semantic dependency is:

$$\begin{aligned} \text{score}(\langle h, m, \mathbf{l} \rangle, x, \mathbf{w}, y') = \\ \text{syntactic_score}(h, m, l_{syn}, x, \mathbf{w}) + \\ \text{sem_score}(h, m, l_{sem\ p_1}, \dots, l_{sem\ p_q}, x, \mathbf{w}, y') \end{aligned}$$

The syntactic score is computed as described in the basic model. Finally, the semantic scoring function computes the semantic score as the sum of the semantic scores for each predicate semantic label:

$$\text{sem_score}(h, m, l_{sem\ p_1}, \dots, l_{sem\ p_q}, x, \mathbf{w}, y') = \sum_{l_{sem\ p_i}} \frac{\phi_{sem}(\langle h, m, l_{sem\ p_i} \rangle, x, y') \cdot \mathbf{w}^{(l_{sem\ p_i})}}{q}$$

Note that each sentence x has a different number of predicates q . To avoid an excessive weight of the semantic component in the global score and a bias towards sentences with many predicates, the score is normalized by the number of predicates in the sentence.

Figure 1 shows an example of a sentence fragment with syntactic and semantic dependencies. The three predicates of the sentence are already identified: $\{p_1 = \text{completed}, p_2 = \text{announced}, p_3 = \text{acquisition}\}$. All dependencies are of the form $d = \langle h, m, l_{syn}, l_{sem\ p_1}, l_{sem\ p_2}, l_{sem\ p_3} \rangle$. Note that semantic labels express semantic relations between a modifier and a predicate that can be anywhere in the sentence. The semantic labeling is not restricted to predicates that are the head of the modifier. In this example, the correct output for the dependency *previously-announced* is $h = \text{announced}$, $m = \text{previously}$, $l_{syn} = \text{AMOD}$, $l_{sem\ p_1} = \text{null}$, $l_{sem\ p_2} = \text{AM-TMP}$, $l_{sem\ p_3} = \text{null}$.

The above described factorization allows the parser to simultaneously assign syntactic and semantic labels and also to maximize a joint syntactic-semantic score of the tree. Note that the semantic scoring function ϕ_{sem} extracts features from the modifier, the head and the predicate of the parsed dependencies. The proposed model allows to capture interactions between syntax and semantics not only because the syntactic and semantic scores are combined but also because the semantic scoring function relies on features extracted from

the head-modifier-predicate relations. Thus, the semantic scoring function depends on the syntactic dependency being built, and, in reverse, the semantic score can modify the dependency chosen.

Regarding implementation issues, note that we compute $|L| + |S| \cdot q$ scores to assign $q + 1$ labels to a given dependency. The scores are computed independently for each label. Otherwise, interactions among these labels, would raise the number of possible combined labels to an exponential number, $|L| \cdot |S|^q$, making the exhaustive evaluation infeasible in practice. Also related to efficiency, we apply syntactic and semantic filters in order to reduce the number of score evaluations. In particular, the set of assignable labels is filtered by the POS of the head and modifier (discarding all labels not previously seen in the training corpus between words with the same POS). Another filter removes the core arguments not present in the frame file of each predicate. This strategy allowed us to significantly improve efficiency without any loss in accuracy.

2.5 Postprocess

A simple postprocess assigns the most frequent sense to each identified predicate. Frequencies were computed from the training corpus. Experiments performed combining the best and second output of the joint parser and enforcing domain constraints via ILP (Punyakanok et al., 2004) showed no significant improvements.

3 Experiments and Results

All the experiments reported here were done using the full training corpus, and results are presented on the development set. The number of features used by the syntactic parser is $\sim 177\text{K}$. The joint parser uses $\sim 45\text{K}$ additional features for recognizing semantic dependencies.

Figure 2 shows the learning curves from epoch 1 to 17 for several subsystems and variants. More specifically, it includes LAS performance on syntactic parsing, both for the individual parser and for the syntactic annotation coming from the joint syntactic-semantic parser. For the latter, also the F_1 score on semantic dependencies and global F_1 results are presented. We can observe that the syntactic LAS scores for the syntactic and joint parsers are very similar, showing that there is no loss in syntactic performance when using the joint syntactic-semantic strategy. Overall re-

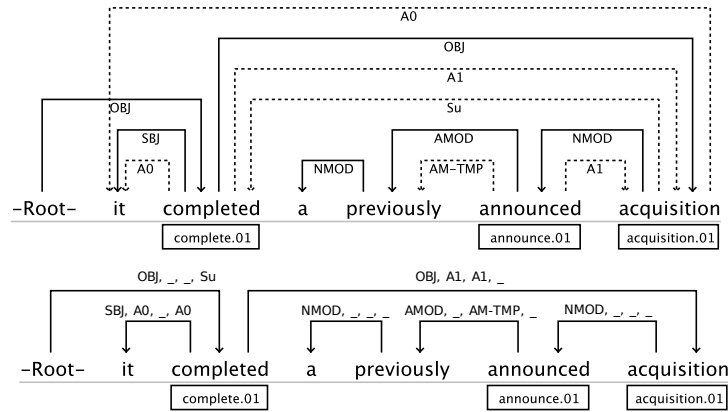


Figure 1: Syntactic and semantic dependencies.

sults are quite stable from epoch 4 (syntax slightly decreases but semantics slightly increases). The overall results on the test set (78.11 global F_1 , 85.84 LAS, 70.35 semantic F_1) were computed by using 5 epochs of training, the optimal on the development set.

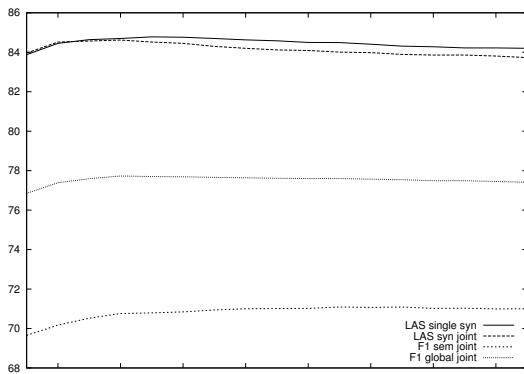


Figure 2: Learning curves for the syntactic-only and joint parsers.

The global F_1 result on the WSJ test corpus is 79.16, but these results drop 9.32 F_1 points on the out-of-domain Brown corpus. Also, a significant performance drop is observed when moving from verb argument classification (74.58 F_1 , WSJ test) to noun argument classification (56.65 F_1 , WSJ test). Note that the same features were used for training noun and verb argument classifiers. These results point out that there is room for improvement on noun argument classification. Finally, a comparison to a simple equivalent pipeline architecture, consisting of applying the syntactic base parser followed by an independent classifica-

tion of semantic dependencies (using exactly the same features) revealed that the joint model outperformed the pipeline by 4.9 F_1 points in the annotation of semantic dependencies.

Regarding efficiency, the proposed architecture is really feasible. About 0.7GB of memory is required for the syntactic parser and 1.5GB for the joint parser. Most of these memory needs are due to the filters used. The filters allowed for a reduction of the computational cost by a factor of 5 with no loss in accuracy. These filters have almost no effect on the theoretical upper bound discarding the correct labels for only 0.2% of the syntactic dependencies and 0.44% of the semantic arguments in the development corpus. The semantic extension of the Eisner algorithm requires only a new table with backpointers for each predicate. Using a single processor of an *amd64 Athlon x2 5000+*, the syntactic parser can be trained at 0.2 s/sentence, and the joint parser at 0.3 s/sentence. Efficiency at test times is only slightly better.

4 Discussion

We have presented a novel joint approach to perform syntactic and semantic parsing by extending Eisner's algorithm. Our model allows to capture syntactic-semantic interactions as the computed syntactic-semantic score is globally optimized. The computational cost of the new setting is admissible in practice, leading to fairly efficient parsers, both in time and memory requirements.

Results obtained with the presented joint approach are promising, though not outstanding in the context of the CoNLL-2008 shared task. We believe that there is room for substantial improvement since many of the current system components

are fairly simple. For instance, higher order extensions to the Eisner algorithm and well-known tricks for dealing with non-projective structures can be incorporated in our model. Also, we plan to incorporate other subtasks in the training of the joint model, such as predicate identification and argument recognition.

One of the potential drawbacks of our current approach is the need for a syntactic parsing preceding the joint model. This previous parse is simply included to permit the extraction of syntax based features. These features (including the syntactic path) could be dynamically computed when performing the joint parsing in the cases in which the predicate coincides with the head of the modifier being processed. These cases account for only 63.6% of the training corpus arguments. If a predicate is located in a sibling sentence span, the dynamic programming algorithm has not yet chosen which of the possible spans will be included in the final parse tree. Also, the predicate can be located at a lower level within the current span. These cases would require to recompute the score of the current span because syntactic path features are not available. The resulting cost would be prohibitive and approximate search needed. Our previous parsing phase is just an efficient and simple solution to the feature extraction problem in the joint model.

As previously seen, the joint model showed a similar syntactic performance and clearly better semantic performance than an equivalent pipeline system, showing that some degree of syntactic-semantic overlap is exploitable. Regarding the former, there is only a moderate degree (63.6%) of direct overlap between the syntactic head-modifier and semantic predicate-modifier relations. If the semantic score is highly dependent on a correct head the resulting increased score could benefit the choosing of a correct dependency. Otherwise, joint scores can introduce a significant amount of noise. All in all, further research is required in this direction.

Acknowledgements

This research has been partially funded by the Spanish Ministry of Education and Science, projects Trangram (TIN2004-07925-C03-02) and OpenMT (TIN2006-15307-C03-02).

References

- Carreras, Xavier, Mihai Surdeanu, and Lluís Màrquez. 2006. Projective dependency parsing with perceptron. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-2006)*.
- Carreras, Xavier. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the 11th Conference on Computational Natural Language Learning (CoNLL-2007)*.
- Collins, Michael. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*.
- Eisner, Jason M. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*.
- Lluís, Xavier. Forthcoming 2008. Joint learning of syntactic and semantic dependencies. Master's thesis, Technical University of Catalonia (UPC).
- McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*.
- Punyakanok, Vasin, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of Coling 2004*.
- Surdeanu, Mihai, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*.
- Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.
- Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP-2004)*.

Collective Semantic Role Labelling with Markov Logic

Sebastian Riedel Ivan Meza-Ruiz

Institute for Communicating and Collaborative Systems

School of Informatics

University of Edinburgh, Scotland

{S.R.Riedel, I.V.Meza-Ruiz}@sms.ed.ac.uk

Abstract

This paper presents our system for the Open Track of the CoNLL 2008 Shared Task (Surdeanu et al., 2008) in Joint Dependency Parsing¹ and Semantic Role Labelling. We use Markov Logic to define a joint SRL model and achieve a semantic F-score of 74.59%, the second best in the Open Track.

1 Introduction

Many SRL systems use a two-stage pipeline that first extracts possible argument candidates (argument identification) and then assigns argument labels to these candidates (argument classification) (Xue and Palmer, 2004). If we also consider the necessary previous step of identifying the predicates and their senses (predicate identification) this yields a three-stage pipeline: predicate identification, argument identification and argument classification.

Our system, on the other hand, follows a joint approach in the spirit of Toutanova et al. (2005) and performs the above steps *collectively*. We decided to use Markov Logic (ML, Richardson and Domingos, 2005), a First Order Probabilistic Language, to develop a global probabilistic model of SRL. By using ML we are able to incorporate the dependencies between the decisions of different stages in the pipeline and the well-known global correlations between the arguments of a predicate (Punyakanok et al., 2005). And since learning

and inference methods were already implemented in the ML software we use, only minimal engineering efforts had to be done.

In contrast to the work of Toutanova et al. (2005) our system applies online learning to train its parameters and exact inference to predict a collective role labelling. Moreover, we jointly label the arguments of *all* predicates in a sentence. This allows us, for example, to require that certain tokens have to be an argument of some predicates in the sentence.

In this paper we also investigate the impact of different levels of interaction between the layers of the joint SRL model. We find that a probabilistic model which resembles a traditional bottom-up pipeline (though jointly trained and globally normalised) performs better than the complete joint model on the WSJ test set and worse on the Brown test set. The worst performance is observed when no interaction between SRL stages is allowed.

In terms of semantic F-score (74.59%) our submitted results are the second best in the Open Track of the Shared Task. Our error analysis indicates that a) the training regime can be improved and b) nominalizations are difficult to handle for the model as it is.

In the next sections we will first briefly introduce Markov Logic. Then we present the Markov Logic model we used in our final submission. We present and analyse our results in section 4 before we conclude in Section 5.

2 Markov Logic

Markov Logic (ML, Richardson and Domingos, 2005) is a Statistical Relational Learning language based on First Order Logic and Markov Networks. It can be seen as a formalism that extends First Order Logic to allow formulae that can be violated with some penalty. From an alternative

©2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

¹Note that in this work we do not consider the parsing task; instead we use the provided dependencies of the open track datasets.

point of view, it is an expressive template language that uses First Order Logic formulae to instantiate Markov Networks of repetitive structure.

Let us describe Markov Logic by considering the predicate identification task. In Markov Logic we can model this task by first introducing a set of logical predicates² such as $isPredicate(Token)$ or $word(Token, Word)$. Then we specify a set of weighted first order formulae that define a distribution over sets of ground atoms of these predicates (or so-called *possible worlds*).

Ideally, the distribution we define with these weighted formulae assigns high probability to possible worlds where SRL predicates are correctly identified and a low probability to worlds where this is not the case. For example, a suitable set of weighted formulae would assign a high probability to the world³

$$\{word(1, Haag), word(2, plays), word(3, Elianti), isPredicate(2)\}$$

and a low one to

$$\{word(1, Haag), word(2, plays), word(3, Elianti), isPredicate(3)\}$$

In Markov Logic a set $M = \{(\phi, w_\phi)\}_\phi$ of weighted first order formulae is called a *Markov Logic Network* (MLN). It assigns the probability

$$p(\mathbf{y}) = \frac{1}{Z} \exp \left(\sum_{(\phi, w) \in M} w \sum_{\mathbf{c} \in C^{n_\phi}} f_{\mathbf{c}}^\phi(\mathbf{y}) \right) \quad (1)$$

to the possible world \mathbf{y} . Here $f_{\mathbf{c}}^\phi$ is a feature function that returns 1 if in the possible world \mathbf{y} the ground formula we get by replacing the free variables in ϕ by the constants in \mathbf{c} is true and 0 otherwise. C^{n_ϕ} is the set of all tuples of constants we can replace the free variables in ϕ with. Z is a normalisation constant. Note that this distribution corresponds to a Markov Network where nodes represent ground atoms and factors represent ground formulae.

For example, if M contains the formula ϕ

$$word(x, 'take') \Rightarrow isPredicate(x)$$

then its corresponding log-linear model has, among others, a feature $f_{t_1}^\phi$ for which x in ϕ has

²In the cases were is not obvious whether we refer to SRL or ML predicates we add the prefix SRL or ML, respectively.

³“Haag plays Elianti” is a segment of a sentence in training corpus.

been replaced by the constant t_1 and that returns 1 if

$$word(1, 'take') \Rightarrow isPredicate(1)$$

is true in \mathbf{y} and 0 otherwise.

We will refer predicates such as $word$ as *observed* because they are known in advance. In contrast, $isPredicate$ is *hidden* because we need to infer it at test time.

2.1 Learning

An MLN we use to model the collective SRL task is presented in section 3. We learn the weights associated this MLN using 1-best MIRA (Crammer and Singer, 2003) Online Learning method.

2.2 Inference

Assuming that we have an MLN, a set of weights and a given sentence then we need to predict the choice of predicates, frame types, arguments and role labels with maximal a posteriori probability. To this end we apply a method that is both exact and efficient: Cutting Plane Inference (CPI, Riedel, 2008) with Integer Linear Programming (ILP) as base solver. We use it for inference at test time as well as during the MIRA online learning process.

3 Model

We define five hidden predicates for the three stages of the task. For predicate identification, we use the predicates $isPredicate$ and $sense$. $isPredicate(p)$ indicates that the word in the position p is an SRL predicate while $sense(p, e)$ signals that predicate in position p has the sense e .

For argument identification, we use the predicates $isArgument$ and $hasRole$. The atom $isArgument(a)$ signals that the word in the position a is a SRL argument of some (unspecified) SRL predicate while $hasRole(p, a)$ indicates that the token at position a is an argument of the predicate in position p .

Finally, for the argument classification stage we define the predicate $role$. Here $role(p, a, r)$ corresponds to the decision that the argument in the position a has the role r with respect to the predicate in the position p .

3.1 Local formulae

We define a set of *local* formulae. A formula is local if its groundings relate any number of observed ground atoms to exactly one hidden ground atom. For example, a grounding of the local formula

$$lemma(p, +l_1) \wedge lemma(a, +l_2) \Rightarrow hasRole(p, a)$$

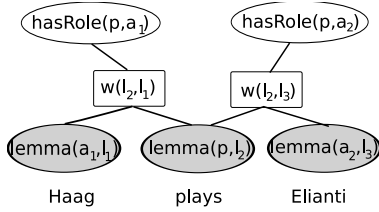


Figure 1: Factor graph for the local formula in section 3.1.

can be seen in the Markov Network of Figure 1. It connects a hidden *hasRole* ground atom to two observed *lemma* ground atoms. Note that the “+” prefix for variables indicates that there is a different weight for each possible pair of lemmas (l_1, l_2) .

For the *hasRole* and *role* predicates we defined local formulae that aimed to reproduce the standard features used in previous work (Xue and Palmer, 2004). This also required us to develop dependency-based versions of the constituent-based features such as the syntactic path between predicate and argument, as proposed by Xue and Palmer (2004).

The remaining hidden predicates, *isPredicate*, *isArgument* and *sense*, have local formulae that relate their ground atoms to properties of a contextual window around the token the atom corresponds to. For this we used the information provided in the closed track training corpus of the shared task (i.e. both versions of lemma and POS tags plus a coarse version of the POS tags).

Instead of describing the local feature set in more detail we refer the reader to our MLN model files.⁴ They can be used both as a reference and as input to our Markov Logic Engine⁵, and thus allow the reader to easily reproduce our results. We believe that this is another advantage of explicitly separating model and algorithms by using first order probabilistic logic languages.

3.2 Global formulae

Global formulae relate several hidden ground atoms. We use them for two purposes: to ensure consistency between the decisions of all SRL stages and to capture some of our intuition about the task. We will refer to formulae that serve the first purpose as *structural constraints*.

For example, a structural constraint is given by

⁴<http://thebeast.googlecode.com/svn/mlns/conll108>

⁵<http://thebeast.googlecode.com>

the (deterministic) formula

$$role(p, a, r) \Rightarrow hasRole(p, a)$$

which ensures that, whenever the argument a is given a label r with respect to the predicate p , this argument must be an argument of a as denoted by *hasRole*(p, a). Note that this formula by itself models the traditional “bottom-up” argument identification and classification pipeline: it is possible to not assign a role r to an predicate-argument pair (p, a) proposed by the identification stage; however, it is impossible to assign a label r to token pairs (p, a) that have not been proposed as potential arguments.

One example of another class of structural constraints is

$$hasRole(p, a) \Rightarrow \exists r. role(p, a, r)$$

which, by itself, models an inverted or “top-down” pipeline. In this architecture the argument classification stage can assign roles to tokens that have not been proposed by the argument identification stage. However, it must assign a label to any token pair the previous stage proposes. Figure 2 illustrates the structural formulae we use in form of a Markov Network.

The formulae we use to ensure consistency between the remaining hidden predicates are omitted for brevity as they are very similar to the bottom-up and top-down formulae we presented above.

For the SRL predicates that perform a labelling task (*role* and *sense*) we also need a structural constraint which ensures that not more than one label is assigned. For instance,

$$(role(p, a, r_1) \wedge r_1 \neq r_2 \Rightarrow \neg role(p, a, r_2))$$

forbids two different semantic roles for a pair of words.

The global formulae that capture our intuition about the task itself can be further divided into two classes. The first one uses deterministic or *hard* constraints such as

$$role(p, a_1, r) \wedge \neg mod(r) \wedge a_1 \neq a_2 \Rightarrow \neg role(p, a_2, r)$$

which forbids cases where distinct arguments of a predicate have the same role unless the role describes a modifier.

The second class of global formulae is *soft* or nondeterministic. For instance, the formula

$$lemma(p, +l) \wedge ppos(a, +p) \wedge hasRole(p, a) \Rightarrow sense(p, +f)$$

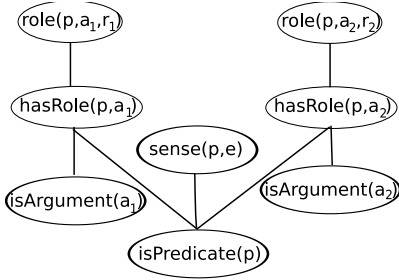


Figure 2: Markov Network that illustrates the structural constraints we use.

is a soft global formula. It captures the observation that the sense of a verb or noun depends on the type of its arguments. Here the type of an argument token is represented by its POS tag.

4 Results

We only submitted results for the Open Track of the Shared Task. Moreover, we focused on SRL and did not infer dependencies; instead we used the MALT dependencies parses provided in the Open Track dataset. Our submission was ranked second out of five with a semantic F1-score of 74.59%.⁶

After submission we also set up additional experiments to evaluate different types and degrees of connectivity between the decisions made by our model. To this end we created four new models: a model that omits top-down structural constraints and thus resembles a (globally trained) bottom-up pipeline (Up); a model that does not contain bottom-up structural constraints and thus resembles a top-down architecture (Down); a model in which stages are not connected at all (Isolated); and finally, a model in which additional global formulae are omitted and the only remaining global formulae are structural (Structural). The results we submitted were generated using the full model (Full).

Table 1 summarises the results for each of these models. We report the F-scores for the WSJ and Brown test corpora provided for the task. In addition we show training and test times for each system.

There are four findings we take from this. First, and somewhat surprisingly, the jointly trained bottom-up model (Up) performs substantially bet-

⁶While we did use information of the open dataset we do believe that it is possible to train a stacked parsing-SRL system that would perform similarly. If so, our system would have the 5th best semantic scores among the 20 participants of the closed track.

Model	WSJ	Brown	Train Time	Test Time
Full	75.72%	65.38%	25h	24m
Up	76.96%	63.86%	11h	14m
Down	73.48%	59.34%	22h	23m
Isolated	60.49%	48.12%	11h	14m
Structural	74.93%	64.23%	22h	33m

Table 1: F-scores for different models.

ter than the full model on the WSJ test corpus. We will try to give an explanation for this result in the next section. Second, the bottom-up model is twice as fast compared to both the full and the top-down model. This is due to the removal of formulae with existential quantifiers that would result in large clique sizes of the ground Markov Network. Third, the isolated model performs extremely poor, particularly for argument classification. Here features defined for the *role* predicate can not make any use of the information in previous stages. Finally, the additional global formulae do improve performance, although not substantially.

4.1 Analysis

A substantial amount of errors in our submitted results (Full) can be attributed to the seemingly random assignment of the very low frequency label “R-AA” (appears once in the training set) to token pairs that should either have a different role or no role at all. Without these false positives, precision would increase by about 1%. Interestingly, this type of error completely disappears for the bottom-up model (Up) and thus seem to be crucial in order understand why this model can outperform the full model.

We believe that this type of error is an artifact of the training regime. For the full model the weights of the *role* predicate only have ensure that the right (true positive) role is the relative winner among all roles. In the bottom-up model they also have to make sure that their cumulative weight is non-negative – otherwise simply not assigning a role r for (p, a) would increase the score even if $hasRole(p, a)$ is predicted with high confidence. Thus more weight is shifted towards the correct roles. This helps the right label to win more likely over the “R-AA” label, whose weights have rarely been touched and are closer to zero.

Likewise, in the bottom-up model the total weight of the *hasRole* features of a wrong (false positive) candidate token pair must be nonpositive. Otherwise picking the wrong candidate would increase overall score and no *role* features can re-

ject this decision because the corresponding structural constraints are missing. Thus more weight is shifted away from false positive candidates, resulting in a higher precision of the *hasRole* predicate. This also means that less wrong candidates are proposed, for which the “R-AA” role is more likely to be picked because its weights have hardly been touched.

However, it seems that by increasing precision in this way, we decrease recall for out-of-domain data. This leads to a lower F1 score for the bottom-up model on the Brown test set.

Another prominent type of errors appear for nominal predicates. Our system only recovers only about 80% of predicates with “NN”, “NNS” and “NNP” tags (and classifies about 90% of these with the right predicate sense). Argument identification and classification performs equally bad. For example, for the “A0” argument of “VB” predicates we get an F-score of 82.00%. For the “A0” of “NN” predicates F-score is 65.92%. The features of our system are essentially taken from the work done on PropBank predicates and we did only little work to adapt these to the case of nominal predicates. Putting more effort into designing features specific to the case of nominal predicates might improve this situation.

5 Conclusion

We presented a Markov Logic Network that jointly performs predicate identification, argument identification and argument classification for SRL. This network achieves the second best semantic F-scores in the Open Track of the CoNLL shared task.

Experimentally we show that results can be further improved by using an MLN that resembles a conventional SRL bottom-up pipeline (but is still jointly trained and globally normalised) instead of a fully connected model. We hypothesise that when training this model more weight is shifted away from wrong argument candidates and more weight is shifted towards correct role labels. This results in higher precision for argument identification and better accuracy for argument classification.

Possible future work includes better treatment of nominal predicates, for which we perform quite poorly. We would also like to investigate the impact of linguistically motivated global formulae more thoroughly. So far our model benefits from them, albeit not substantially.

References

- Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 2003.
- V. Punyakanok, D. Roth, and W. Yih. Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Annual Conference on Computational Natural Language Learning*, 2005.
- Matthew Richardson and Pedro Domingos. Markov logic networks. Technical report, University of Washington, 2005.
- Sebastian Riedel. Improving the accuracy and efficiency of map inference for markov logic. In *Proceedings of the Annual Conference on Uncertainty in AI*, 2008.
- Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*, 2008.
- Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 2005.
- Nianwen Xue and Martha Palmer. Calibrating features for semantic role labeling. In *Proceedings of the Annual Conference on Empirical Methods in Natural Language Processing*, 2004.

Hybrid Learning of Dependency Structures from Heterogeneous Linguistic Resources

Yi Zhang

Language Technology Lab
DFKI GmbH
yzhang@coli.uni-sb.de

Rui Wang

Computational Linguistics
Saarland University, Germany
rwang@coli.uni-sb.de

Hans Uszkoreit

Language Technology Lab
DFKI GmbH
uszkoreit@dfki.de

Abstract

In this paper we present our syntactic and semantic dependency parsing system participated in both closed and open competitions of the CoNLL 2008 Shared Task. By combining the outcome of two state-of-the-art syntactic dependency parsers, we achieved high accuracy in syntactic dependencies (87.32%). With MRSeS from grammar-based HPSG parsers, we achieved significant performance improvement on semantic role labeling (from 71.31% to 71.89%), especially in the out-domain evaluation (from 60.16% to 62.11%).

1 Introduction

The CoNLL 2008 shared task (Surdeanu et al., 2008) provides a unique chance of comparing different syntactic and semantic parsing techniques in one unified open competition. Our contribution in this joint exercise focuses on the combination of different algorithms and resources, aiming not only for state-of-the-art performance in the competition, but also for the dissemination of the learnt lessons to related sub-fields in computational linguistics.

The so-called hybrid approach we take has two folds of meaning. For syntactic dependency parsing, we build our system based on state-of-the-art algorithms. Past CoNLL share task results have shown that transition-based and graph-based algorithms started from radically different ideas, yet achieved largely comparable results. One of the questions we would like to investigate is whether the

combination of the two approaches on the output level leads to even better results.

For the semantic role labeling (SRL) task, we would like to build a system that allows us to test the contribution of different linguistic resources. To our special interest is to examine the deep linguistic parsing systems based on hand-crafted grammars. During the past decades, various large scale linguistic grammars have been built, some of which achieved both broad coverage and high precision. In combination with other advances in deep linguistic processing, e.g. efficient parsing algorithms, statistical disambiguation models and robust processing techniques, several systems have reached mature stage to be deployed in applications. Unfortunately, due to the difficulties in cross-framework evaluation, fair comparison of these systems with state-of-the-art data-driven statistical parsers is still hard to achieve. More importantly, it is not even clear whether deep linguistic analysis is necessary at all for tasks such as shallow semantic parsing (also known as SRL). Drawing a conclusion on this latter point with experiments using latest deep parsing techniques is one of our objectives.

The remainder of the paper is structured as follows. Section 2 introduces the overall system architecture. Section 3 explains the voting mechanism used in the syntactic parser. Section 4 describes in detail the semantic role labeling component. Section 5 presents evaluation results and error analysis. Section 6 concludes the paper.

2 System Architecture

As shown in Figure 1, our system is a two-stage pipeline. For the syntactic dependencies, we apply two state-of-the-art dependency parsers and combine their results based on a voting model. For

© 2008. Licensed under the *Creative Commons Attribution-NonCommercial-Share Alike 3.0 Unported License* (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

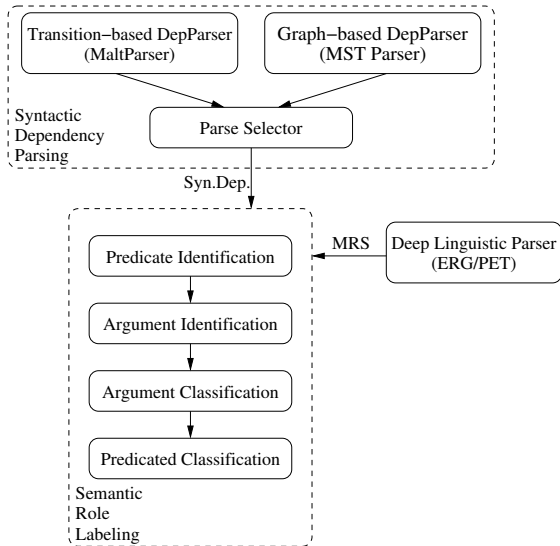


Figure 1: System Architecture

the semantic roles, we extracted features from the previous stage, combined with deep parsing results (in MRS), and use statistical classification models to make predictions. In particular, the second part can be further divided into four stages: predicate identification (PI), argument identification (AI), argument classification (AC), and predicate classification (PC). Maximum entropy-based machine learning techniques are used in both components which we will see in detail in the following sections.

3 Syntactic Dependency Parsing

For obtaining syntactic dependencies, we have combined the results of two state-of-the-art dependency parsers: the MST parser (McDonald et al., 2005) and the MaltParser (Nivre et al., 2007).

The MST parser formalizes dependency parsing as searching for maximum spanning trees (MSTs) in directed graphs. A major advantage of their framework is the ability to naturally and efficiently model both projective and non-projective parses. To learn these structures they used online large-margin learning that empirically provides state-of-the-art performance.

The MaltParser is a transition-based incremental dependency parser, which is language-independent and data-driven. It contains a deterministic algorithm, which can be viewed as a variant of the basic shift-reduce algorithm. The learning method they applied is support vector machine and experimental evaluation confirms that the MaltParser can achieve robust, efficient and accurate parsing for a

wide range of languages.

Since both their parsing algorithms and machine learning methods are quite different, we decide to take advantages of them. After a comparison between the results of the two parsers¹, we find that,

1. The MST parser is better at the whole structure. In several sentences, the MaltParser was wrong at the root node, but the MST parser is correct.
2. The MaltParser is better at some dependency labels (e.g. TMP, LOC, etc.).

These findings motivate us to do a voting based on both outputs. The features considered in the voting model are as follows:

- **Dependency path:** two categories of dependency paths are considered as features: 1) the POS-Dep-POS style and 2) the Dep-Dep style. The former consists of part-of-speech (POS) tags and dependency relations appearing in turns; and the latter only contains dependency relations. The maximum length of the dependency path is three dependency relations.
- **Root attachments:** the number of tokens attached to the ROOT node by the parser in one sentence
- **Sentence length:** the number of tokens in each input sentence
- **Projectivity:** whether the parse is projective or not

With these features, we apply a statistical model to predict, for each sentence, we choose the parsing result from which parser. The voted result will be our syntactic dependency output and be passed to the later stages.

4 Semantic Role Labeling

4.1 Overview

The semantic role labeling component of our system is comprised of a pipeline model with four

¹In this experiment, we use second order features and the projective decoder for the MST parser trained with 10 iterations, and Arc-eager algorithm with a quadric polynomial kernel for the MaltParser.

sub-components that performs predicate identification (PI), argument identification (AI), argument classification (AC) and predicate classification (PC) respectively. The output in previous steps are taken as input information to the following stages. All these components are essentially based on a maximum entropy statistical classifier, although with different task-specific optimizations and feature configurations in each step. Depending on the available information from the input data structure, the same architecture is used for both closed and open challenge runs, with different feature types. Note that our system does not make use of or predict SU chains.

Predicate Identification The component makes binary prediction on each input token whether it forms a predicate in the input sentence. This predictor precedes other components because it is a relatively easy task (comparing to the following components). Also, making this prediction early helps to cut down the search space in the following steps. Based on the observation on the training data, we limit the PI predictor to only predict for tokens with certain POS types (POSeS marked as predicates for at least 50 times in the training set). This helps to significantly improve the system efficiency in both training and prediction time without sacrificing prediction accuracy.

It should be noted that the prediction of nominal predicates are generally much more difficult (based on CoNLL 2008 shared task annotation). The PI model achieved 96.32 F-score on WSJ with verbal predicates, but only 84.74 on nominal ones.

Argument Identification After PI, the arguments to the predicted predicates are identified with the AI component. Similar to the approach taken in Hacioglu (2004), we use a statistical classifier to select from a set of candidate nodes in a dependency tree. However, instead of selecting from a set of neighboring nodes from the predicate word², we define the concept of argument path as a chain of dependency relations from the predicate to the argument in the dependency tree. For instance, an argument path [SBJ | TMP] indicates that if the predicate is syntactically depending as SBJ on a node which has a TMP child, then the TMP node

²Hacioglu (2004) defines a tree-structured *family* of a predicate as a measure of locality. It is a set of dependency relation nodes that consists of the predicate's parent, children, grandchildren, siblings, siblings' children and siblings' grandchildren with respect to its dependency tree

(sibling to the predicate) is an argument candidate. While Hacioglu (2004)'s approach focus mainly on local arguments (with respect to the syntactic dependencies), our approach is more suitable of capturing long distance arguments from the predicate. Another minor difference is that we allow predicate word to be its own argument (which is frequently the case for nominal predicates) with an empty argument path [|].

The set of effective argument paths are obtained from the training set, sorted and filtered according to their frequencies, and used in testing to obtain the candidate arguments. By setting a frequency threshold, we are able to select the most useful argument paths. The lower the threshold is, the higher coverage one might get in finding candidate arguments, accompanied with a higher average candidate number per predicate and potentially a more difficult task for the statistical classifier. By experimenting with different frequency thresholds on the training set, we established a frequency threshold of 40, which guarantees candidate argument coverage of 95%, and on average 5.76 candidates per predicate. Given that for the training set each predicate takes on average 2.13 arguments, the binary classifier will have relatively balanced prediction classes.

Argument Classification For each identified argument, an argument label will be assigned during the argument classification step. Unlike the binary classifiers in previous two steps, AC uses a multi-class classifier that predicts from the inventory of argument labels. For efficiency reasons, we only concern the most frequent 25 argument labels.

Predicate Classification The final step in the SRL component labels the predicted predicate with a predicate name. Due to the lack of lexical resources in the closed competition, this step is scheduled for the last, in order to benefit from the predictions made in the previous steps. Unlike the previous steps, the statistical model used in this step is a ranking model. We obtained a list of candidate frames and corresponding rolesets from the provided PropBank and NomBank data. Each predicted predicate is mapped onto the potential rolesets it may take. When the frame for the predicate word is missing from the list, or there is only one candidate roleset for it, the predicate name is assigned deterministically (word stem concatenated with “.01” for frame missing predicates, the unam-

biguous roleset name when there is only one candidate). When there are more than one candidate rolesets, a ranking model is trained to select the most probable roleset for a given predicate given the syntactic and semantic context.

4.2 Features

The feature types used in our SRL component are summarized in Table 1, with the configurations of our submitted “closed” and “open” runs marked. Numerous different configurations with these feature types have been experimented on training and development data. The results show that feature types 1–14 are the best performing ones. Features related to the siblings of the predicate only introduce minor performance variation. We also find the named entity labels does not lead to immediate improvement of SRL performance. The WordNet sense feature does achieve minor performance increase on PI and PC, although the significant remains to be further examined. Based on the pipeline model, we find it difficult to achieve further improvement by incorporate more features types from provided annotation. And the variance of SRL performance with different open features is usually less than 1%. To clearly show the contribution of extra external resources, these less contributing features (siblings, named entity labels and WordNet sense) are not used in our submitted “open” runs.

MRSes as features to SRL As a novel point of our SRL system, we incorporate parsing results from a linguistic grammar-based parsing system in our “open” competition run. In this experiment, we used English Resource Grammar (ERG, Flickinger (2000)), a precision HPSG grammar for English. For parsing, we used PET (Callmeier, 2001), an efficient HPSG parser, together with extended partial parsing module (Zhang et al., 2007) for maximum robustness. The grammar is hand-crafted and the disambiguation models are trained on Redwoods treebanks. They present general linguistic knowledge, and are not tuned for the specific domains in this competition.

While the syntactic analysis of the HPSG grammar is largely different from the dependency annotation used in this shared task, the semantic representations do share a similar view on predicate-argument structures. ERG uses as its semantic representation the Minimal Recursion Semantics (MRS, Copestake et al. (2005)), a non-recursive flat

structure that is suitable for underspecifying scope ambiguities. A predicate-argument backbone of MRS can be extracted by identifying shared variables between elementary predications (EPS).

In order to align the HPSG parser’s I/O with CoNLL’s annotation, extensive mapping scripts are developed to preprocess the texts, and extract backbone from output MRSes. The unknown word handling techniques (Zhang and Kordoni, 2005) are used to overcome lexical gaps. Only the best analysis is used for MRS extraction. Without partial parsing, the ERG achieves around 70% of raw coverage on the training data. When partial parsing is used, almost all the sentences received either full or partial analysis (except for several cases where computational resources are exhausted), and the SRL performance improves by $\sim 0.5\%$.

5 Results

Among 20+ participant groups, our system ranked seventh in the “closed” competition, and first in the “open” challenge. The performance of the syntactic and semantic components of our system are summarized in Table 2.

		In-Domain		Out-Domain	
		Lab.	Unlab.	Lab.	Unlab.
Syntactic Dep.		88.14%	90.78%	80.80%	86.12%
SRL	Closed	72.67%	82.68%	60.16%	76.98%
	Open	73.08%	83.04%	62.11%	78.48%

Table 2: Labeled and unlabeled attachment scores in syntactic dependency parsing and F1 score for semantic role labeling.

The syntactic voting and semantic labeling parts of our system are implemented in Java together with a few Perl scripts. Using the open source TADM for parameter estimation, our the voting component take no more than 1 minute to train and 10 seconds to run (on WSJ testset). The SRL component takes about 1 hour for training, and no more than 30 seconds for labeling (WSJ testset).

Result analysis shows that the combination of the two state-of-the-art parsers delivers good syntactic dependencies (ranked 2nd). Error analysis shows most of the errors are related to prepositions. One category is the syntactic ambiguity of pp-attachment, e.g. in “when trading was halted in Philip Morris”, “in” can be attached to either “trading” or “halted”. The other category is the LOC and TMP tags in phrases like “at the end of the day”, “at the point of departure”, etc.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	P lemma	P POS	P rel	P-parent POS	A lemma	A POS	A rel	P-children POSes	P-children rel	P-A path	A-children POSes	A-children rels	P precedes A?	A's position	P-siblings POSes	P-siblings rels	P NE label	P WN sense	P MRS EP-name	P MRS-args labels	P MRS-args POSes	A MRS EP-name	A MRS-preds labels	A MRS-preds POSes
PI	⊗	⊗	⊗	⊗				⊗	⊗						×	×				○	○	○		
AI	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗		×	×			○	○	○	○	○	○
AC	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	×	×			○	○	○	○	○	○
PC	⊗	⊗	⊗	⊗				⊗	⊗						×	×			○	○	○			

Table 1: Feature types used in semantic role labeling sub-components. Feature types marked with × are used in the “closed” run; feature types marked with ○ are used in the “open” run; feature types marked with ⊗ are used in both runs. P denotes predicate; A denotes semantic argument.

The results on semantic role labeling show, sometimes, even with syntactic errors of LOC/TMP tags, the semantic role labeler can still predict AM-LOC/AM-TMP correctly, which indicates the robustness of our hybrid approach. By comparing our “closed” and “open” runs, the MRS features do introduce a clear performance improvement. The performance gain is even more significant in out-domain test, showing that the MRS features from ERG are indeed much less domain dependent. Another example worth mentioning is that, in the sentence “Scotty regarded the ear and the grizzled hair around it with a moment of interest”, it is extremely difficult to know that “Scotty” is a semantic role of “interest”.

Also, we are the only group that submitted runs for both tracks, and achieved better performance in open competition. Although the best ways of integrating deep linguistic processing techniques remain as an open question, the achieved results at least show that hand-crafted grammars like ERG do provide heterogeneous linguistic insights that can potentially find their usage in data-driven NLP tasks as such.

6 Conclusion

In this paper, we described our hybrid system on both syntactic and semantic dependencies labeling. We built a voting model to combine the results of two state-of-the-art syntactic dependency parsers, and a pipeline model to combine deep parsing results for SRL. The experimental results showed the advantages of our hybrid strategy, especially on the cross-domain data set. Although the optimal ways of combining deep processing techniques remains to be explored, the

performance gain achieved by incorporating hand-crafted grammar outputs shows a promising direction of study for both fields.

References

- Callmeier, Ulrich. 2001. Efficient parsing with large-scale unification grammars. Master’s thesis, Universität des Saarlandes, Saarbrücken, Germany.
- Copestake, Ann, Dan Flickinger, Carl J. Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: an introduction. *Research on Language and Computation*, 3(4):281–332.
- Flickinger, Dan. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.
- Hacioglu, Kadri. 2004. Semantic role labeling using dependency trees. In *Proceedings of COLING 2004*, pages 1273–1276, Geneva, Switzerland, Aug 23–Aug 27.
- McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT-EMNLP 2005*, pages 523–530, Vancouver, Canada.
- Nivre, Joakim, Jens Nilsson, Johan Hall, Atanas Chaney, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(1):1–41.
- Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*, Manchester, UK.
- Zhang, Yi and Valia Kordoni. 2005. A statistical approach towards unknown word type prediction for deep grammars. In *Proceedings of the Australasian Language Technology Workshop 2005*, pages 24–31, Sydney, Australia.
- Zhang, Yi, Valia Kordoni, and Erin Fitzgerald. 2007. Partial parse selection for robust deep processing. In *Proceedings of ACL 2007 Workshop on Deep Linguistic Processing*, pages 128–135, Prague, Czech.

Parsing Syntactic and Semantic Dependencies with Two Single-Stage Maximum Entropy Models *

Hai Zhao and Chunyu Kit

Department of Chinese, Translation and Linguistics
City University of Hong Kong
83 Tat Chee Avenue, Kowloon, Hong Kong, China
haizhao@cityu.edu.hk, ctckit@cityu.edu.hk

Abstract

This paper describes our system to carry out the joint parsing of syntactic and semantic dependencies for our participation in the shared task of CoNLL-2008. We illustrate that both syntactic parsing and semantic parsing can be transformed into a word-pair classification problem and implemented as a single-stage system with the aid of maximum entropy modeling. Our system ranks the fourth in the closed track for the task with the following performance on the WSJ+Brown test set: 81.44% labeled macro F1 for the overall task, 86.66% labeled attachment for syntactic dependencies, and 76.16% labeled F1 for semantic dependencies.

1 Introduction

The joint parsing of syntactic and semantic dependencies introduced by the shared task of CoNLL-08 is more complicated than syntactic dependency parsing or semantic role labeling alone (Surdeanu et al., 2008). For semantic parsing, in particular, a dependency-based representation is given but the predicates involved are unknown, and we also have nominal predicates besides the verbal ones. All these bring about more difficulties for learning. This paper presents our research for participation in the CoNLL-2008 shared task, with a highlight on our strategy to select learning framework and features for maximum entropy learning.

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

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported license* (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

The rest of the paper is organized as follows. The next section presents the technical details of our system and Section 3 its evaluation results. Section 4 looks into a few issues concerning our forthcoming work for this shared task, and Section 5 concludes the paper.

2 System Description

For the sake of efficiency, we opt for the maximum entropy model with Gaussian prior as our learning model for both the syntactic and semantic dependency parsing. Our implementation of the model adopts L-BFGS algorithm for parameter optimization as usual (Liu and Nocedal, 1989). No additional feature selection techniques are applied.

Our system consists of three components to deal with syntactic and semantic dependency parsing and word sense determination, respectively. Both parsing is formulated as a single-stage word-pair classification problem, and the latter is carried out by a search through the NomBank (Meyers et al., 2004) or the PropBank (Palmer et al., 2005)¹.

2.1 Syntactic Dependency Parsing

We use a shift-reduce scheme to implement syntactic dependency parsing as in (Nivre, 2003). It takes a step-wised, history- or transition-based approach. It is basically a word-by-word method with a projective constraint. In each step, the classifier checks a word pair, e.g., *TOP*, the top of a stack for processed words, and, *NEXT*, the first word in the unprocessed word sequence, in order to determine if a dependent label should be assigned to them. Besides two arc-building actions, a shift action and a reduce action are also defined to meet the projective constraint, as follows.

¹These two dictionaries that we used are downloaded from CoNLL-2008 official website.

Notation	Meaning
s	Clique in the top of stack
s_{-1}, \dots	The first clique below the top of stack, etc.
i, i_{+1}, \dots	The first (second) clique in the unprocessed sequence, etc.
$dprel$	Dependent label
h	Head
lm	Leftmost child
rm	Rightmost child
rn	Right nearest child
$form$	Word form
$lemma$	Word lemma
pos	Predicted PoS tag
sp_Y	Split Y , which may be $form$, $lemma$ or pos .
\cdot	's, e.g., ' $s.dprel$ ' means dependent label of the clique in the top of stack
$/$	Feature combination, i.e., ' $s.pos/i.pos$ ' means $s.pos$ and $i.pos$ together as a feature function.
p	The current predicate candidate
a	The current argument candidate

Table 1: Feature Notations

1. **Left-arc:** Add an arc from $NEXT$ to TOP and pop the stack.
2. **Right-arc:** Add an arc from TOP to $NEXT$ and push $NEXT$ onto the stack.
3. **Reduce:** Pop TOP from the stack.
4. **Shift:** Push $NEXT$ onto the stack.

We implement a left-to-right arc-eager parsing model in a way that the parser scan through an input sequence from left to right and the right dependents are attached to their heads as soon as possible (Hall et al., 2007). To construct a single-stage system, we extend the left-/right-arc actions to their correspondent multi-label actions as necessary. Including 32 left-arc and 66 right-arc actions, altogether a 100-class problem is yielded for the parsing action classification for this shared task.

Since only projective sequences can be handled by the shift-reduce scheme, we apply the pseudo-projective transformation introduced by (Nivre and Nilsson, 2005) to projectivize those non-projective sequences. Our statistics show that only 7.6% sequences and less than 1% dependencies in the corpus provided for training are non-projective. Thus, we use a simplified strategy to projectivize an input sequence. Firstly, we simply replace the head of a non-projective dependency by its original head's head but without any additional dependent label encoding for the purpose of deprojectivizing the output during decoding. Secondly, if the above standard projectivization step cannot eliminate all

Basic	Extension
$x.sp_Y$	itself, its previous two and next two Y s, and all bigrams within the five-clique window, (x is s or i , and Y is $form$, $lemma$ or pos .)
$x.Y$	(x is s or i , and Y is $form$, $lemma$ or pos .)
$x.Y/i.Y$	(x is s or s_{-1} and Y is pos , sp_lemma or sp_pos)
-	$s.h.sp_form$
-	$s.dprel$
-	$s.lm.dprel$
-	$s.rn.dprel$
-	$i.lm.sp_pos$
-	$s.lm.dprel/s.dprel$
-	$s.lm.sp_pos/s.sp_pos$
-	$s.h.sp_pos/s.sp_pos$
-	$x.sp_pos rootscore$ (x is s or i .)
-	$s.sp_pos/i.sp_pos pairscore$
-	$s.curroot.sp_pos/i.sp_pos$

Table 2: Features for Syntactic Parsing

non-projective dependencies in a sequence, then the word with the shortest sequence (rather than dependent tree) distance to the original head will be chosen as the head of a non-projective dependency. In practice, the above two-step projectivization procedure can eliminate all non-projective dependencies in all sequences. Our purpose here is to provide as much data as possible for training, and only projective sequences are input for training and output for decoding.

While memory-based and margin-based learning approaches such as support vector machines are popularly applied to shift-reduce parsing, our work provides evidence that the maximum entropy model can achieve a comparative performance with the aid of a suitable feature set. With feature notations in Table 1, we use a feature set as shown in Table 2 for syntactic parsing.

Here, we explain ' $rootscore$ ', ' $pairscore$ ' and $curroot$ in Table 2. Both $rootscore$ and $pairscore$ return the log frequency for an event in the training corpus. The former counts a given split PoS occurring as ROOT, and the latter two split PoS's combination associated with a dependency label. The feature $curroot$ returns the root of a partial parsing tree that includes a specified node.

2.2 Semantic Dependency Parsing

Assuming no predicates overtly known, we keep using a word-pair classifier to perform semantic parsing through a single-stage processing. Specifically, we specify the first word in a word pair as a predicate candidate (i.e., a semantic head, and noted as p in our feature representation) and the next as an argument candidate (i.e., a semantic de-

Basic	Extension
$x.sp_Y$	itself, its previous and next cliques, and all bigrams within the three-clique window. (Y is <i>form</i> or <i>lemma</i> .) ^a
$x.sp_pos$	itself, its previous and next two cliques, and all bigrams within the five-clique window.
$x.Y$	(Y is <i>form</i> , <i>lemma</i> or <i>pos</i> .)
$p.Y/i.Y$	(Y is <i>sp_lemma</i> or <i>sp_pos</i> .)
-	a is the same as p
-	$x.is_Verb_or_Noun$
-	<i>bankAdvice</i>
- ^b	$a.h.sp_form$
-	$x.dprel$
-	$x.lm.dprel$
-	$p.rm.dprel$
-	$p.lm.sp_pos$
-	$a.lm.dprel/a.dprel$
-	$a.lm.sp_pos/a.sp_pos$
-	$a.sp_Y/a.dprel$ (Y is <i>lemma</i> or <i>pos</i> .)
-	$x.sp_lemma/x.h.sp_form$
-	$p.sp_lemma/p.h.sp_pos$
-	$p.sp_pos/p.dprel$
-	$a.preddir$ ^c
-	$p.voice/a.preddir$ ^d
-	$x.posSeq$ ^e
-	$x.dprelSeq$ ^f
-	$a.dpTreeLevel$ ^g
-	$a/p dpRelation$
-	$a/p SharedPosPath$
-	$a/p SharedDprelPath$
-	$a/p x.posPath$
-	$a/p x.dprelPath$
-	$a/p dprelPath$

^a x is p or a throughout the whole table.

^bThis and the following features are all concerned with a known syntactic dependency tree.

^c*preddir*: the direction to the current predicate candidate.

^d*voice*: if the syntactic head of p is *be* and p is not ended with *-ing*, then p is passive.

^e*posSeq*: PoS tag sequence of all syntactic children

^f*dprelSeq*: syntactic dependent label sequence of all syntactic children

^g*dpTreeLevel*: the level in the syntactic parse tree, counted from the leaf node.

Table 3: Features for Semantic Parsing

pendent, and noted as a). We do not differentiate between nominal and verbal predicates and our system handles them in exactly the same way. If decoding outputs show that no arguments can be found for a predicate candidate in the decoding output, then this candidate will be naturally discarded from the output predicate list.

When no constraint available, however, all word pairs in the an input sequence must be considered, leading to very poor efficiency in computation for no gain in effectiveness. Thus, the training sample needs to be pruned properly.

For predicate, only nouns and verbs are considered possible candidates. That is, all words without a split PoS in these two categories are filtered

out. Many prepositions are also marked as predicate in the training corpus, but their arguments' roles are 'SU', which are not counted the official evaluation.

For argument, a dependency version of the pruning algorithm in (Xue and Palmer, 2004) is used to find, in an iterative way, the current syntactic head and its siblings in a parse tree in a constituent-based representation. In this representation, the head of a phrase governs all its sisters in the tree, as illustrated in the conversion of constituents to dependencies in (Lin, 1995). In our implementation, the following equivalent algorithm is applied to select argument candidates from a syntactic dependency parse tree.

Initialization: Set the given predicate candidate as the current node;

- (1) The current node and all of its syntactic children are selected as argument candidates.
- (2) Reset the current node to its syntactic head and repeat step (1) until the root is reached.

This algorithm can cover 98.5% arguments while reducing about 60% of the training samples, according to our statistics. However, this is achieved at the price of including a syntactic parse tree as part of the input for semantic parsing.

The feature set listed in Table 3 is adopted for our semantic parsing, some of which are borrowed from (Hacioglu, 2004). Among them, *dpTreeRelation* returns the relationship of a and p in a syntactic parse tree. Its possible values include *parent*, *sibling*, *child*, *uncle*, *grand parent* etc. Note that there is always a path to the ROOT in the syntactic parse tree for either a or p . Along the common part of these two paths, *SharedDprelPath* returns the sequence of dependent labels collected from each node, and *SharedPosPath* returns the corresponding sequence of PoS tags. *x.dprelPath* and *x.posPath* return the PoS tag sequence from x to the beginnings of *SharedDprelPath* and *SharedPosPath*, respectively. $a/p|dprelPath$ returns the concatenation of $a.dprelPath$ and $p.dprelPath$.

We may have an example to show how the feature *bankAdvice* works. Firstly, the current processed semantic role labels and argument candidate direction are checked. Specifically, they are the arguments $A0$ and $A1$ that have been marked before the predicate candidate p and the current argument identification direction after p . Secondly,

	UAS	LAS	Label-Acc.
Development	88.78	85.85	91.14
WSJ	89.86	87.52	92.47
Brown	85.03	79.83	86.71
WSJ+Brown	89.32	86.66	91.83

Table 4: The Results of Syntactic Parsing (%)

	Data	Precision	Recall	F-score
Label.	Development	79.76	72.25	75.82
	WSJ	80.57	74.97	77.67
	Brown	66.28	61.29	63.69
	WSJ+Brown	79.03	73.49	76.16
Unlab.	Development	89.58	81.15	85.16
	WSJ	89.48	83.26	86.26
	Brown	83.14	76.88	79.89
	WSJ+Brown	88.79	82.57	85.57

Table 5: The Results of Semantic Parsing (%)

each example² of p in NomBank or PropBank that depends on the split PoS tag of p is checked if it partially matches the current processed role labels. If a unique example exists in this form, e.g., `Before:A0-A1; After:A3`, then this feature returns `A3` as feature value. If no matched or multiple matched examples exist, then this feature returns a default value.

2.3 Word Sense Determination

The shared task of CoNLL-2008 for word sense disambiguation task is to determine the sense of an output predicate. Our system carries out this task by searching for a right example in the given NomBank or PropBank. The semantic role set scheme of each example for an output predicate is checked. If a scheme is found to match the output semantic role set of a predicate, then the corresponding sense for the first match is chosen; otherwise the system outputs ‘01’ as the default sense.

3 Evaluation Results

Our evaluation is carried out on a 64-bit ubuntu Linux installed server with double dual-core AMD Opteron processors of 2.8GHz and 8GB memory. The full training set for CoNLL-2008 is used to train the maximum entropy model. The training for the syntactic parser costs about 200 hours and

²The term “example” means a chunk in NomBank or PropBank, which demonstrates how semantic roles occur around a specified predicate. For example, for a sense item of the predicate `access` in PropBank, we first have `<arg n="0">a computer</arg><rel>access</rel><arg n="1">its memory</arg>`, and then a role set scheme for this sense as `Before:A0;After:A1`.

	Data	Precision	Recall	F-score
Label. Macro	Development	82.80	79.05	80.88
	WSJ	84.05	81.25	82.62
	Brown	73.05	70.56	71.78
	WSJ+Brown	82.85	80.08	81.44
Unlab. Macro	Development	89.18	84.97	87.02
	WSJ	89.67	86.56	88.09
	Brown	84.08	80.96	82.49
	WSJ+Brown	89.06	85.94	87.47
Label. Micro	Development	83.69	80.71	82.17
	WSJ	85.07	82.88	83.96
	Brown	75.14	73.09	74.10
	WSJ+Brown	83.98	81.80	82.88
Unlab. Micro	Development	89.06	85.90	87.45
	WSJ	89.72	87.42	88.56
	Brown	84.38	82.07	83.21
	WSJ+Brown	89.14	86.83	87.97

Table 6: Overall Scores (%)

4.1GB memory and that for the semantic parser costs about 170 hours and 4.9GB memory. The running time in each case is the sum of all running time for all threads involved. When a parallel optimization technique is applied to speedup the training, the time can be reduced to about 1/3.5 of the above.

The official evaluation results for our system are presented in Tables 4, 5 and 6. Following the official guideline of CoNLL-2008, we use unlabeled attachment score (UAS), labeled attachment score (LAS) and label accuracy to assess the performance of syntactic dependency parsing. For semantic parsing, the unlabeled scores metric the identification performance and the labeled scores the overall performance of semantic labeling.

4 To Do

Although we are unable to follow our plan to do more than what we have done for this shared task, because of the inadequate computational resource and limited time, we have a number of techniques in our anticipation to bring in further performance improvement.

While expecting to accomplish the joint inference of syntactic and semantic parsing, we only have time to complete a system with the former to enhance the latter. But we did have experiments in the early stage of our work to show that a syntactic dependency parser can make use of available semantic dependency information to enhance its performance by 0.5-1%³.

Most errors in our syntactic parsing are related

³We used the outputs of a semantic parser, either predicted or gold-standard, as features for syntactic parsing.

to the dependencies of comma and prepositions. We need to take care of them, for PP attachment is also crucial to the success of semantic parsing. Extra effort is paid, as illustrated in previous work such as (Xue and Palmer, 2004), to handle such cases, especially when a PP is involved. We find in our data that about 1% arguments occur as a grandchild of a predicate through PP attachment.

Syntactic parsing contributes crucially to the overall performance of the joint parsing by providing a solid basis for further semantic parsing. Thus there is reason to believe that improvement of syntactic dependency parsing can be more influential than that of semantic parsing to the overall improvement. Only one model was used for syntactic parsing in our system, in contrast to the existing work using an ensemble technique for further performance enhancement, e.g., (Hall et al., 2007). Again, the latter means much more computational cost should be taken.

Though it was not done before submission deadline, we also tried to enhance the semantic parsing with some more sophisticated inputs from the syntactic parsing. One is predicted syntactic parsed tree input that may be created by cross-validation rather than the gold-standard syntactic input that our submitted semantic parser was actually trained on. Another is the n -best outputs of the syntactic parser. However, only the single-best output of the syntactic parser was actually used.

5 Conclusion

As presented in the above sections, our system to participate in the CoNLL-2008 shared task is implemented as two single-stage maximum entropy learning. We have tackled both syntactic and semantic parsing as a word-pair classification problem. Despite the simplicity of this approach, our system has produced promising results.

Acknowledgements

We wish to thank Dr. Wenliang Chen of NICT, Japan for helpful discussions on dependency parsing, and two anonymous reviewers for their valuable comments.

References

Hacioglu, Kadri. 2004. Semantic role labeling using dependency trees. In *Proceedings of the 20th*

international conference on Computational Linguistics (COLING-2004), pages 1273–1276, Geneva, Switzerland, August 23rd-27th.

Hall, Johan, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? a study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 933–939, Prague, Czech, June.

Lin, Dekang. 1995. A dependency-based method for evaluating broad-coverage parser. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1420–1425, Montréal, Québec, Canada, August 20-25.

Liu, Dong C. and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528.

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

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL-2005)*, pages 99–106, Ann Arbor, Michigan, USA, June 25-30.

Nivre, Joakim. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160, Nancy, France, April 23-25.

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

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*, pages 88–94, Barcelona, Spain, July 25-26.

A Combined Memory-Based Semantic Role Labeler of English

Roser Morante, Walter Daelemans, Vincent Van Asch

CNTS - Language Technology Group

University of Antwerp

Prinsstraat 13, B-2000 Antwerpen, Belgium

{Roser.Morante,Walter.Daelemans,Vincent.VanAsch}@ua.ac.be

Abstract

We describe the system submitted to the closed challenge of the CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. Syntactic dependencies are processed with the Malt-Parser 0.4. Semantic dependencies are processed with a combination of memory-based classifiers. The system achieves 78.43 labeled macro F1 for the complete problem, 86.07 labeled attachment score for syntactic dependencies, and 70.51 labeled F1 for semantic dependencies.

1 Introduction

In this paper we describe the system submitted to the closed challenge of the CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies (Surdeanu et al., 2008). Compared to the previous shared tasks on semantic role labeling, the innovative feature of this one is that it consists of extracting both syntactic and semantic dependencies. The semantic dependencies task comprises labeling the semantic roles of nouns and verbs and disambiguating the frame of predicates.

The system that we present extracts syntactic and semantic dependencies independently. Syntactic dependencies are processed with the Malt-Parser 0.4 (Nivre, 2006; Nivre et al., 2007). Semantic dependencies are processed with a combination of memory-based classifiers.

Memory-based language processing (Daelemans and van den Bosch, 2005) is based on the

idea that NLP problems can be solved by storing solved examples of the problem in their literal form in memory, and applying similarity-based reasoning on these examples in order to solve new ones. Keeping literal forms in memory has been argued to provide a key advantage over abstracting methods in NLP that ignore exceptions and sub-regularities (Daelemans et al., 1999).

Memory-based algorithms have been previously applied to semantic role labeling. Van den Bosch et al. (2004) participated in the CoNLL-2004 shared task with a system that extended the basic memory-based learning method with class n-grams, iterative classifier stacking, and automatic output post-processing. Tjong Kim Sang et al. (2005) participated in the CoNLL-2005 shared task with a system that incorporates spelling error correction techniques. Morante and Busser (2007) participated in the SemEval-2007 competition with a semantic role labeler for Spanish based on gold standard constituent syntax. These systems use different types of constituent syntax (shallow parsing, full parsing). We are aware of two systems that perform semantic role labeling based on dependency syntax previous to the CoNLL-2008 shared task. Hacioglu (2004) converts the data from the CoNLL-2004 shared task into dependency trees and uses support vector machines. Morante (2008) describes a memory-based semantic role labeling system for Spanish based on gold standard dependency syntax.

We developed a memory-based system for the CoNLL-2008 shared task in order to evaluate the performance of this methodology in a completely new semantic role labeling setting.

The paper is organised as follows. In Section 2 the system is described, Section 3 contains an analysis of the results, and Section 4 puts forward some

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

conclusions.

2 System description

The system processes syntactic and semantic dependencies independently. The syntactic dependencies are processed with the MaltParser 0.4. The semantic dependencies are processed with a cascade of memory-based classifiers. We use the IB1 classifier 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 IB1, similarity is defined by computing (weighted) overlap of the feature values of a test instance and a memorized example. The metric combines a per-feature value distance metric with global feature weights that account for relative differences in discriminative power of the features.

2.1 Syntactic dependencies

The MaltParser 0.4¹ (Nivre, 2006; Nivre et al., 2007) is an inductive dependency parser that uses four essential components: a deterministic algorithm for building labeled projective dependency graphs; history-based feature models for predicting the next parser action; support vector machines for mapping histories to parser actions; and graph transformations for recovering non-projective structures.

The learner type used was support vector machines, with the same parameter options reported by (Nivre et al., 2006). The parser algorithm used was Nivre, with the options and model (eng.par) for English as specified on <http://w3.msi.vxu.se/users/jha/conll07/>. The tagset.pos, tagset.cpos and tagset.dep were extracted from the training corpus.

2.2 Semantic dependencies

The semantics task consists of finding the predicates, assigning a PropBank or a NomBank frame to them and extracting their semantic role dependencies. Because of lack of resources, we did not have time to develop a word sense disambiguation system. So, predicates were assigned the frame ‘.01’ by default.

The system handles the semantic role labeling task in three steps: predicate identification, seman-

tic dependency classification, and combination of classifiers.

2.2.1 Predicate identification

In this phase, a classifier predicts if a word is a predicate or not. The IB1 algorithm was parameterised by using overlap as the similarity metric, information gain for feature weighting, using 7 k -nearest neighbors, and weighting the class vote of neighbors as a function of their inverse linear distance. The instances represent all nouns and verbs in the corpus and they have the following features:

- Word form, lemma, part of speech (POS), the three last letters of the word, and the lemma and POS of the five previous and five next words. To obtain the previous word we perform a linear left-to-right search. This is how *previous* has to be interpreted further on when features are described.

The accuracy of the classifier on the development test is 0.9599 (4240/4417) for verbs and 0.8981 (9226/10272) for nouns.

2.2.2 Semantic dependency classification

In this phase, three groups of multi-class classifiers predict in one step if there is a dependency between a word and a predicate, and the type of dependency, i.e. semantic role.

Group 1 (G1) consists of two classifiers: one for predicates that are nouns and another for predicates that are verbs. The instances represent a predicate-word combination. The predicates are those that have been classified as such in the previous phase. As for the combining words, determiners and certain combinations are excluded based on the fact that they never have a role in the training corpus.

The IB1 algorithm was parameterised by using overlap as the similarity metric, information gain for feature weighting, using 11 k -nearest neighbors, and weighting the class vote of neighbors as a function of their inverse linear distance. The features of the noun classifier are:

- About the predicate: word form. About the combining word: word form, POS, dependency type, word form of the two previous and two next words. Chain of POS types between the word and the predicate. Distance between the word and the predicate. Binary feature indicating if the word depends on the predicate. Six chains of POS tags between the word and its three previous and three next predicates in relation to the current predicate.

¹Web page of MaltParser 0.4: <http://w3.msi.vxu.se/~nivre/research/MaltParser.html>.

The features of the verb classifier are:

- The same as for the noun classifier and additionally: POS of the word next to the current combining word, binary feature indicating if the combining word depends on the predicate previous to the current predicate, binary feature indicating if the predicate previous to the combining word is located before or after the current predicate.

The verb classifier achieves an overall accuracy of 0.9244 (80805/87412), and the noun classifier, 0.9173 (69836/76132) in the development set.

Group 2 (G2) consists also of two classifiers: one for predicates that are nouns and another for predicates that are verbs. The instances represent combinations of word-predicate, but the test corpus contains only those instances that G1 has classified as having a role.

The IB1 algorithm was parameterised in the same way as for G1, except that it computes 7 k -nearest neighbors instead of 11. The two classifiers use the same features:

- About the predicate: word form, chain of lemmas of the syntactic siblings, chain of lemmas of the syntactic children. About the combining word: word form, POS, dependency type, word form of the two previous and the two next words, POS+type of dependency and lemma of the syntactic father, chain of dependency types and chain of lemmas of the syntactic children. Chain of POS types between word and predicate, distance and syntactic dependency type between word and predicate.

The verb classifier achieves an overall accuracy of 0.5656 (4160/7355), and the noun classifier, 0.5017 (2234/4452) in the development set.

Group 3 (G3) consists of one classifier. Like G2, instances represent combinations of word-predicate, but the test corpus contains only those instances that G1 has classified as having a role.. The IB1 algorithm was parameterised in the same way as for G2. It uses the following features:

About the predicate: lemma, POS, POS of the 3 previous and 3 next predicates. About the combining word: lemma, POS, and dependency type, POS of the 3 previous and 3 next words. Distance between the predicate and the word. A binary feature indicating if the combining word is located before or after the predicate.

The classifier achieves an overall accuracy of 0.5527 (6526/11807).

2.2.3 Combination of classifiers

In this phase the three groups of classifiers are combined in a simple way: if G2 and G3 agree in classifying a semantic dependency, their solution is chosen, else the solution of G1 is chosen. This system combination choice is explained by the fact that G1 has a higher accuracy than G2 and G3 when the three classifiers are applied to the development set. G2 and G3 are used to eliminate overgeneration of roles by G1.

The performance of the system in the development corpus with only the G1 classifiers is 66.07 labeled F1. The combined system achieves a 10.8% error reduction, with 69.75 labeled F1.

3 Results

The results of the system are shown in Table 1. We will focus on commenting on the semantic scores. The system scores 71.88 labeled F1 in the in-domain corpus (WSJ) and 59.23 in the out-of-domain corpus (Brown). Unlabeled F1 in the WSJ corpus is almost 10% higher than labeled F1. Labeled precision is 12.40% higher than labeled recall.

	WSJ	BROWN
SYNTACTIC SCORES		
Labeled attachment score	86.88	79.58
Unlabeled attachment score	89.37	84.85
Label accuracy score	91.48	86.00
SEMANTIC SCORES		
Labeled precision	78.61	65.25
Labeled recall	66.21	54.23
Labeled F1	71.88	59.23
Unlabeled precision	89.13	83.61
Unlabeled recall	75.08	69.48
Unlabeled F1	81.50	75.89
OVERALL MACRO SCORES		
Labeled macro precision	82.74	72.41
Labeled macro recall	76.54	66.90
Labeled macro F1	79.52	69.55
Unlabeled macro precision	89.25	84.23
Unlabeled macro recall	82.22	77.16
Unlabeled macro F1	85.59	80.54

Table 1: Results of the system in the WSJ and BROWN corpora expressed in %.

3.1 Discussion

The performance of the semantic role labeler is affected considerably by the performance of the first classifier for predicate detection. The system cannot recover from the predicates that are missed in this phase. Experiments without the first classifier and with gold standard predicates (detection and classification) result in 80.89 labeled F1, 9.01 %

higher than the results of the system with predicate detection. We opted for identifying predicates as a first step in order to reduce the number of training instances for the second phase, classification of semantic dependencies. For the same reason, we opted for selecting only nouns and verbs as instances, aware of the fact that we would miss a very low number of predicates with other categories. The results of predicate identification can be improved by setting up a combined system, instead of a single classifier, and by incorporating a system for frame disambiguation.

Equally important would be to find better features for the identification of noun predicates, since the features used generalise better for verbs than for nouns. Table 2 shows that the system is better at identifying verbs than it is at identifying nouns.

	Total	F1 Pred. Id.&Cl.	F1 Pred. Id.
CC	3	-	-
CD	1	-	-
IN	3	-	-
JJ	16	-	-
NN	3635	77.57	85.59
NNP	10	30.77	38.46
NNS	1648	75.47	83.65
PDT	2	-	-
RP	4	-	-
VB	1278	79.28	98.87
VBD	1320	85.44	99.24
VBG	742	77.05	94.41
VBN	985	76.43	92.08
VBP	343	78.60	97.81
VBZ	504	80.94	97.36
WP	2	-	-
WRB	2	-	-

Table 2: Predicate (Pred.) identification (Id.) and classification (Cl.) in the WSJ corpus expressed in %.

A characteristic of the semantic role labeler is that recall is considerably lower than precision (12.40 %). This can be further analysed with the data shown in Table 3.

Except for the dependency VB*+AM-NEG, precision is higher than recall for all semantic dependencies. We run the semantic role labeler with gold standard predicates and with gold standard syntax and predicates. The difference between precision and recall is around 10 % in both cases, which confirms that low recall is a characteristic of the semantic role labeler, probably caused by the fact that the features do not generalise good enough. The semantic role labeler with gold stan-

Dependency	Total	Recall	Prec.	F1
NN*+A0	2339	42.41	77.80	54.90
NN*+A1	3757	61.17	78.32	68.69
NN*+A2	1537	45.48	82.24	58.57
NN*+A3	349	50.14	88.38	63.98
NN*+AM-ADV	32	9.38	37.50	15.01
NN*+AM-EXT	33	18.18	85.71	30.00
NN*+AM-LOC	232	30.60	63.96	41.40
NN*+AM-MNR	344	34.59	79.87	48.27
NN*+AM-NEG	35	2.86	100.00	5.56
NN*+AM-TMP	492	54.88	83.33	66.18
VB*+A0	3509	68.99	82.63	75.20
VB*+A1	4844	74.24	83.28	78.50
VB*+A2	1085	55.94	69.21	61.87
VB*+A3	169	41.42	79.55	54.48
VB*+A4	99	74.75	88.10	80.88
VB*+AM-ADV	488	38.93	59.19	46.97
VB*+AM-CAU	70	50.00	70.00	58.33
VB*+AM-DIR	81	29.63	57.14	39.02
VB*+AM-DIS	315	52.70	74.11	61.60
VB*+AM-EXT	32	50.00	59.26	54.24
VB*+AM-LOC	355	52.11	57.10	54.49
VB*+AM-MNR	335	46.57	61.18	52.88
VB*+AM-MOD	539	92.21	95.95	94.04
VB*+AM-NEG	227	94.71	90.34	92.47
VB*+AM-PNC	113	33.63	54.29	41.53
VB*+AM-TMP	1068	64.51	80.40	71.58
VB*+C-A1	192	65.10	74.85	69.64
VB*+R-A0	222	65.77	87.43	75.07
VB*+R-A1	155	49.68	73.33	59.23
VB*+R-AM-LOC	21	23.81	71.43	35.71
VB*+R-AM-TMP	52	46.15	66.67	54.54

Table 3: Semantic dependencies identification and classification in the WSJ corpus for dependencies with more than 20 occurrences expressed in %.

dard predicates scores 86.06 % labeled precision and 76.32 % labeled recall. The semantic role labeler with gold standard predicates and syntax scores 89.20 % precision and 79.47 % recall.

Table 3 also shows that the unbalance between precision and recall is higher for dependencies of nouns than for dependencies of verbs, and that both recall and precision are higher for dependencies from verbs. Thus, the system performs better for verbs than for nouns. This is in part caused by the fact that more noun predicates than verb predicates are missed in the predicate identification phase. The scores of the the semantic role labeler with gold standard predicates show lower differences in F1 between verbs and nouns.

The fact that the semantic role labeler performs 3.16 % labeled F1 better with gold standard syntax (compared to the system with gold standard syntax and predicates) confirms that gold standard syntax provides useful information to the system.

Additionally, the difference in performance between the semantic role labeler presented to the

competition and the semantic role labeler with gold standard predicates (9.01 % labeled F1) suggests that, although the results of the system are encouraging, there is room for improvement, and improvement should focus on increasing the recall scores.

4 Conclusions

In this paper we have presented a system submitted to the closed challenge of the CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. We have focused on describing the part of the system that extracts semantic dependencies, a combination of memory-based classifiers. The system achieves a semantic score of 71,88 labeled F1. Results show that the system is considerably affected by the first phase of predicate identification, that the system is better at extracting the semantic dependencies of verbs than those of nouns, and that recall is substantially lower than precision. These facts suggest that, although the results are encouraging, there is room for improvement.

5 Acknowledgements

This work was made possible through financial support from the University of Antwerp (GOA project BIOGRAPH), and from the Flemish Institute for the Promotion of Innovation by Science and Technology Flanders (IWT) (TETRA project GRAVITAL). The experiments were carried out in the CalcUA computing facilities. We are grateful to Stefan Becuwe for his support.

References

- Cover, T. M. and P. E. Hart. 1967. Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, 13:21–27.
- Daelemans, W. and A. van den Bosch. 2005. *Memory-based language processing*. Cambridge University Press, Cambridge, UK.
- Daelemans, W., A. Van den Bosch, and J. Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning*, 34:11–41.
- Daelemans, W., 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.
- Hacioglu, K. 2004. Semantic role labeling using dependency trees. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, Morristown, NJ, USA. ACL.
- Morante, R. and B. Busser. 2007. ILK2: Semantic role labelling for Catalan and Spanish using TiMBL. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 183–186.
- Morante, R. 2008. Semantic role labeling tools trained on the Cast3LB-CoNLL-SemRol corpus. In *Proceedings of the LREC 2008*, Marrakech, Morocco.
- Nivre, J., J. Hall, J. Nilsson, G. Eryigit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning, CoNLL-X*, New York City, NY, June.
- Nivre, J., J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007. Malt-Parser: a language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.
- Nivre, J. 2006. *Inductive Dependency Parsing*. Springer.
- Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.
- Tjong Kim Sang, E., S. Canisius, A. van den Bosch, and T. Bogers. 2005. Applying spelling error correction techniques for improving semantic role labelling. In *Proceedings of the Ninth Conference on Natural Language Learning (CoNLL-2005)*, Ann Arbor, MI.
- van den Bosch, A., S. Canisius, W. Daelemans, I. Hendrickx, and E. Tjong Kim Sang. 2004. Memory-based semantic role labeling: Optimizing features, algorithm, and output. In Ng, H.T. and E. Riloff, editors, *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, Boston, MA, USA.

A Puristic Approach for Joint Dependency Parsing and Semantic Role Labeling

Alexander Volokh

LT-lab, DFKI
66123 Saarbrücken, Germany

Alexander.Volokh@dfki.de

Günter Neumann

LT-lab, DFKI
66123 Saarbrücken, Germany

neumann@dfki.de

Abstract

We present a puristic approach for combining dependency parsing and semantic role labeling. In a first step, a data-driven strict incremental deterministic parser is used to compute a single syntactic dependency structure using a MEM trained on the syntactic part of the CoNLL 2008 training corpus. In a second step, a cascade of MEMs is used to identify predicates, and, for each found predicate, to identify its arguments and their types. All the MEMs used here are trained only with labeled data from the CoNLL 2008 corpus. We participated in the closed challenge, and obtained a labeled macro F1 for WSJ+Brown of 19.93 (20.13 on WSJ only, 18.14 on Brown). For the syntactic dependencies we got similar bad results (WSJ+Brown=16.25, WSJ=16.22, Brown=16.47), as well as for the semantic dependencies (WSJ+Brown=22.36, WSJ=22.86, Brown=17.94). The current results of the experiments suggest that our risky puristic approach of following a strict incremental parsing approach together with the closed data-driven perspective of a joined syntactic and semantic labeling was actually too optimistic and eventually too puristic.

The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies (cf. Surdeanu, 2008) offered to us an opportunity to initiate, implement and test new ideas on large-scale data-driven incremental dependency parsing. The topic and papers of the ACL-2004 workshop “Incremental Parsing: Bringing Engi-

neering and Cognition Together” (accessible at <http://aclweb.org/anthology-new/W/W04/#0300>) present a good recent overview into the field of incremental processing from both an engineering and cognitive point of view.

Our particular interest is the exploration and development of strict incremental deterministic strategies as a means for fast data-driven dependency parsing of large-scale online natural language processing. By strict incremental processing we mean, that the parser receives a stream of words w_1 to w_n word by word in left to right order, and that the parser only has information about the current word w_i , and the previous words w_1 to w_{i-1} .¹ By deterministic processing we mean that the parser has to decide immediately and uniquely whether and how to integrate the newly observed word w_i with the already constructed (partial) dependency structure without the possibility of revising its decision at later stages. The strategy is data-driven in the sense that the parsing decisions are made on basis of a statistical language model, which is trained on the syntactic part of the CoNLL 2008 training corpus. The whole parsing strategy is based on Nivre (2007), but modifies it in several ways, see sec. 2 for details.

Note that there are other approaches of incremental deterministic dependency parsing that assume that the complete input string of a sentence is already given before parsing starts and that this additional right contextual information is also used as a feature source for language modeling, e.g., Nivre (2007).

In light of the CoNLL 2008 shared task, this actually means that, e.g., part-of-speech tagging and lemmatization has already been performed

¹ Note that in a truly strict incremental processing regime the input to the NLP system is actually a stream of signals where even the sentence segmentation is not known in advance. Since in our current system, the parser receives a sentence as given input, we are less strict as we could be.

for the complete sentence before incremental parsing starts, so that this richer source of information is available for defining the feature space. Since, important word-based information especially for a dependency analysis is already known for the whole sentence before parsing starts, and actually heavily used during parsing, one might wonder, what the benefit of such a weak incremental parsing approach is compared to a non-incremental approach. Since, we thought that such an incremental processing perspective is a bit too wide (especially when considering the rich input of the CoNLL 2008 shared task), we wanted to explore a strict incremental strategy.

Semantic role labeling is considered as a post-process that is applied on the output of the syntactic parser. Following Hacioglu (2004), we consider the labeling of semantic roles as a classification problem of dependency relations into one of several semantic roles. However, instead of post-processing a dependency tree firstly into a sequence of relations, as done by Hacioglu (2004), we apply a cascade of statistical models on the unmodified dependency tree in order to identify predicates, and, for each found predicate, to identify its arguments and their types. All the language models used here are trained only with labeled data from the CoNLL 2008 corpus; cf. sec. 3 for more details.

Both, the syntactic parser and the semantic classifier are language independent in the sense that only information contained in the given training corpus is used (e.g., PoS tags, dependency labels, information about direction etc.), but no language specific features, e.g., no PropBank frames nor any other external language and knowledge specific sources.

The complete system has been designed and implemented from scratch after the announcement of the CoNLL 2008 shared task. The main goal of our participation was therefore actually on being able to create some initial software implementation and baseline experimentations as a starting point for further research in the area of data-driven incremental deterministic parsing.

In the rest of this brief report, we will describe some more details of the syntactic and semantic component in the next two sections, followed by a description and discussion of the achieved results.

1 Syntactic Parsing

Our syntactic dependency parser is a variant of the incremental non-projective dependency parser described in Nivre (2007). Nivres' parser is incremental in the sense, that although the complete list of words of a sentence is known, construction of the dependency tree is performed strictly from left to right. It uses Treebank-induced classifiers to deterministically predict the actions of the parser. The classifiers are trained using support vector machines (SVM). A further interesting property of the parser is its capability to derive (a subset of) non-projective structures directly. The core idea here is to exploit a function $permissible(i, j, d)$ that returns true if and only if the dependency links $i \rightarrow j$ and $j \rightarrow i$ have a degree less than or equal to d given the dependency graph built so far. A degree $d=0$ gives strictly projective parsing, while setting $d=\infty$ gives unrestricted non-projective parsing; cf. Nivre (2007) for more details. The goal of this function is to restrict the call of a function $link(i, j)$ which is a nondeterministic operation that adds the arc $i \rightarrow j$, the arc $j \rightarrow i$, or does nothing at all. Thus the smaller the value of d is the fewer links can be drawn.

The function $link(i, j)$ is directed by a trained SVM classifier that takes as input the feature representation of the dependency tree built so far and the (complete) input $x = w_1, \dots, w_n$ and outputs a decision for choosing exactly one of the three possible operations.

We have modified Nivres algorithm as follows:

1. Instead of using classifiers learned by SVM, we are using classifiers based on Maximum Entropy Models (MEMs), cf. (Manning and Schütze, 1999).²
2. Instead of using the complete input x , we only use the prefix from w_1 up to the current word w_i . In this way, we are able to model a stricter incremental processing regime.
3. We are using a subset of feature set described in Nivre (2007).³ In particular, we had to discard all features from Nivre's set that refer to a word right to the current word in order to retain our

² We are using the `opennlp.maxent` package available via <http://maxent.sourceforge.net/>.

³ We mean here all features that are explicitly described in Nivre (2007). He also mentions the use of some additional language specific features, but they are not further described, and, hence not known to us.

strict incremental behavior. Additionally, we added the following features:

- a. Has j more children in the current dependency graph compared with the average number of children of element of same POS.
- b. Analogously for node i
- c. Distance between i and j

Although some results – for example Wang et al. (2006) – suggest that SVMs are actually more suitable for deterministic parsing strategies than MEMs, we used MEMs instead of SVM basically for practical reasons: 1) we already had hands-on experience with MEMs, 2) training time was much faster than SVM, and 3) the theoretical basis of MEMs should give us enough flexibility for testing with different sets of features.

Initial experiments applied on the same corpora as used by Nivre (2007), soon showed that our initial prototype is certainly not competitive in its current form. For example, our best result on the TIGER Treebank of German (Brants et al., 2002) is 53.6% (labeled accuracy), where Nivre reports 85.90%; cf. Volokh (2008) and sec. 4 for more details

Anyway, we decided to use it as a basis for the CoNLL 2008 shared task and to combine it with a component for semantic role labeling at least to get some indication of “what went wrong”.

2 Semantic Role Labeling

On the one hand, it is clear that we should expect that our current version of the strict incremental deterministic parsing regime still returns too erroneous dependency analysis. On the other hand, we decided to apply semantic role labeling on the parser’s output. Hence, the focus was set towards a robust strictly data-driven approach.

Semantic role labeling is modeled as a sequence of classifiers that follow the structure of predicates, i.e., firstly candidate predicates are identified and then the arguments are looked up.

Predicate and argument identification both proceed in two steps: first determine whether a word can be a predicate or argument (or not), and then, each found predicate (argument) is typed. More precisely, semantic role labeling receives the output of the syntactic parser and performs the following steps in that order:

1. Classify each word as being a predicate or not using a MEM-based classifier.
2. Assign to each predicate its reading. Currently, this is done on basis of the fre-

quency readings as determined from the corpus (for unknown words, we simply assign the reading .01 to the lemma if the whole word was classified as a predicate).

3. For each predicate identified in a sentence, classify each word as argument for this predicate or not using a MEM-based classifier.
4. For each argument identified for each predicate, assign its semantic role using a MEM-based classifier.

For step 1 the following features are used for word w_i : 1) word form, 2) word lemma, 3) POS, 4) dependency type, 5) number of dependent elements in subtree of w_i , 6) POS of parent, 7) dependency type of parent, 8) children or parent of word belong to prepositions, and 9) parent is predicate.

For step 3 the same features are used as in step 1, but 5) (for arguments the number of children is not important) and two additional features are used: 10) left/right of predicate (arguments are often to the right of its predicate), and 11) distance to predicate (arguments are not far from the predicate). Finally, for step 4 the same features are used as in step 1, but 5) and 9).

3 Experiments

As mentioned above, we started the development of the system from scratch with a very small team (actually only one programmer). Therefore we wanted to focus on certain aspects, totally abandoning our claims for achieving decent results for the others. One of our major goals was the construction of correct syntactic trees and the recognition of the predicate-argument structure - a subtask which mainly corresponds to the unlabeled accuracy. For that reason we reduced the scale of our experiments concerning such steps as dependency relation labeling, determining the correct reading for the predicates or the proper type of the arguments.

Unfortunately only the labeled accuracy was evaluated at this year’s task, which was very frustrating in the end.

3.1 Syntactic Dependencies

For testing the strict incremental dependency parser we used the CoNLL 2008 shared task training and development set. Our best syntactic score that we could achieve on the development data was merely unlabeled attachment score (UAL) of 45.31%. However, as mentioned in sec. 2, we used a set of features proposed by Nivre,

which contains 5 features relying on the dependency types. Since we couldn't develop a good working module for this part of the task due to the lack of time, we couldn't exploit these features.

Note that for this experiment and all others reported below, we used the default settings of the `opennlp MEM` trainer. In particular this means that 100 iterations were used in all training runs and that for all experiments no tuning of parameters and smoothing was done, basically because we had no time left to exploit it in a sensible way. These parts will surely be revised and improved in the future.

3.2 Semantic Dependencies

As we describe in the sec. 3 our semantic module consists of 4 steps. For the first step we achieve the F-score of 76.9%. Whereas the verb predicates are recognized very well (average score for every verb category is almost 90%), we do badly with the noun categories. Since our semantic module depends on the input produced by the syntactic parser, and is influenced by its errors, we also did a test assuming a 100% correct parse. In this scenario we could achieve the F-score of 79.4%.

We have completely neglected the second step of the semantic task. We didn't even try to do the feature engineering and to train a model for this assignment, basically because of time constraints. Neither did we try to include some information about the predicate-argument structure in order to do better on this part of the task. The simple assignment of the statistically most frequent reading for each predicate reduced the accuracy from 76.9% down to 69.3%. In case of perfect syntactic parse the result went down from 79.4% to 71.5%.

Unfortunately the evaluation software doesn't provide the differentiation between the unlabeled and labeled argument recognition, which corresponds to our third and fourth steps respectively. Whereas we put some effort on identifying the arguments, we didn't focus on their classification. Therefore the overall best labeled attachment score for our system is 29.38%, whereas the unlabeled score is 50.74%. Assuming the perfect parse the labeled score is 32.67% and the unlabeled score is 66.73%. In our further work we will try to reduce this great deviation between both results.

3.3 Runtime performance

One of the main strong sides of the strict incremental approach is its runtime performance.

Since we are restricted in our feature selection to the already seen space to the left of the current word, both the training and the application of our strategy are done fast.

The training of our MEMs for the syntactic part requires 62 minutes. The training of the models for our semantic components needs 31 minutes. The test run of our system for the test data from the Brown corpus (425 sentences with 7207 tokens) lasted 1 minute and 18 seconds. The application on the WSJ test data (2399 sentences with 57676 tokens) took 20 minutes and 42 seconds. The experiments have been performed on a computer with one Intel Pentium 1,86 Ghz processor and 1GB memory.

4 Results and Discussion

The results of running our current version on the CoNLL 2008 shared task test data were actually a knockdown blow. We participated in the closed challenge, and obtained for the complete problem a labeled macro F1 for WSJ+Brown of 19.93 (20.13 on WSJ only, 18.14 on Brown). For the syntactic dependencies we got similar bad results (WSJ+Brown = 16.25, WSJ = 16.22, Brown = 16.47), as well as for the semantic dependencies (WSJ+Brown = 22.36, WSJ = 22.86, Brown = 17.94).

We see at least the following two reasons for this disastrous result: On the one hand we focused on the construction of correct syntactic trees and the recognition of the predicate-argument structure which were only parts of the task. On the other hand we stuck to our strict incremental approach, which greatly restricted the scope of development of our system.

Whereas the labeling part, which was so far considerably neglected, will surely be improved in the future, the strict incremental strategy in its current form will probably have to be revised.

4.1 Post-evaluation experiments

We have already started beginning the improvement of our parsing system, and we briefly discuss our current findings. On the technical level we already found a software bug that at least partially might explain the unexpected high difference in performance between the results obtained for the development set and the test set. Correcting this error now yields an UAL of 53.45% and an LAL of 26.95% on the syntactic

part of the Brown test data which is a LAL-improvement of about 10%.

On the methodological level we are studying the effects of relaxing some of the assumptions of our strict incremental parsing strategy. In order to do so, we developed a separate model for predicting the unlabeled edges and a separate model for labeling them. In both cases we used the same features as described in sec. 2, but added features that used a right-context in order to take into account the PoS-tag of the N-next words viz. N=5 for the syntactic parser and N=3 for the labeling case. Using both models during parsing interleaved, we obtained UAL=65.17% and LAL=28.47% on the development set.

We assumed that the low LAL might have been caused by a too narrow syntactic context. In order to test this assumption, we decoupled the prediction of the unlabeled edges and their labeling, such that the determination of the edge labels is performed *after* the complete unlabeled dependency tree is computed. Labeling of the dependency edges is then simply performed by running through the constructed parse trees assigning each edge the most probable dependency type. This two-phase strategy achieved an LAL of 60.44% on the development set, which means an improvement of about 43%. Applying the two-phase parser on the WSJ test data resulted in UAL=65.22% and LAL=62.83%; applying it on the Brown test data resulted in UAL=66.50% and LAL=61.11%, respectively.

Of course, these results are far from being optimal. Thus, beside testing and improving our parser on the technical level, we will run further experiments for different context sizes, exploiting different settings of parameters of the classifier and feature values, and eventually testing other ML approaches. The focus here will be on the development of unlabeled edge models, because it seems that an improvement here is substantial for an overall improvement. For example, applying the decoupled edge labeling model directly on the given unlabeled dependency trees of the development set (i.e. we assume an UAL of 100%) gave as an LAL of 92.88%.

Beside this, we will also re-investigate interleaved strategies of unlabeled edge and edge labeling prediction as a basis for (mildly-) strict incremental parsing. Here, it might be useful to relax the strict linear control regime by exploring beam search strategies, e.g. along the lines of Collins and Roark (2004).

5 Conclusion

We have presented a puristic approach for joint dependency parsing and semantic role labeling. Since, the development of our approach has been started from scratch, we didn't manage to deal with all problems. Our focus was on setting up a workable backbone, and then on trying to do as much feature engineering as possible. Our bad results on the CoNLL 2008 suggest that our current strategy was a bit too optimistic and risky, and that the strict incremental deterministic parsing regime seemed to have failed in its current form. We are now in the process of analysis of "what went wrong", and have already indicated some issues in the paper.

Acknowledgement

The work presented here was partially supported by a research grant from the German Federal Ministry of Education, Science, Research and Technology (BMBF) to the DFKI project HyLaP, (FKZ: 01 IW F02). We thank the developers of the Opennlp.maxent software package.

References

- Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank in Proceedings of the Workshop on Treebanks and Linguistic Theories Sozopol.
- Collins, Michael, and Brian Roark. (2004). Incremental Parsing with the Perceptron Algorithm. ACL 2004.
- Hacioglu, Kadri. 2004. Semantic Role Labeling Using Dependency Trees. *Coling 2004*.
- Nivre, Joakim. 2007. Incremental Non-Projective Dependency Parsing. *NAACL-HLT 2007*.
- Manning, Christopher, and Hinrich Schütze. 1999. Foundations of statistical natural language processing. Cambridge, Mass.: MIT Press.
- Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.
- Volokh, Alexander. 2008. Datenbasiertes Dependenzparsing. *Bachelor Thesis, Saarland University*.
- Wang, Mengqui, Kenji Sagae, and Teruko Mitamura. 2006. A Fast, Accurate Deterministic Parser for Chinese. ACL 2006.

Discriminative Learning of Syntactic and Semantic Dependencies

Lu Li¹, Shixi Fan², Xuan Wang¹, Xiaolong Wang¹
Shenzhen Graduate School, Harbin Institute of Technology,
Xili, Shenzhen 518055, China
¹{lli, wangxuan, wangxl}@insun.hit.edu.cn
²fanshixi@hit.edu.cn

Abstract

A Maximum Entropy Model based system for discriminative learning of syntactic and semantic dependencies submitted to the CoNLL-2008 shared task (Surdeanu, et al., 2008) is presented in this paper. The system converts the dependency learning task to classification issues and reconstructs the dependent relations based on classification results. Finally F1 scores of 86.69, 69.95 and 78.35 are obtained for syntactic dependencies, semantic dependencies and the whole system respectively in closed challenge. For open challenge the corresponding F1 scores are 86.69, 68.99 and 77.84.

1 Introduction

Given sentences and corresponding part-of-speech of each word, the learning of syntactic and semantic dependency contains two separable goals: (1) building a dependency tree that defines the syntactic dependency relationships between separated words; (2) specifying predicates (no matter verbs or nouns) of the sentences and labeling the semantic dependents for each predicate.

In this paper a discriminative parser is proposed to implement maximum entropy (ME) models (Berger, et al., 1996) to address the learning task. The system is divided into two main subsystems: syntactic dependency parsing and semantic dependency labeling. The former is used to find a well-formed syntactic dependency tree that occupies all the words in the sentence. If edges are added between any two words, a full-connected

graph is constructed and the dependency tree could be found using a maximum spanning tree (MST) algorithm (McDonald, et al., 2005). The latter focuses on separable predicates whose semantic dependents could be determined using classification tools, such as ME models¹ etc..

We participated in both closed and open challenge of the CoNLL-2008 shared task (Surdeanu, et al., 2008). Results are reported on both development and test sets in this paper.

2 System Description

2.1 Syntactic Parsing

The goal of syntactic parsing is to create a labeled syntactic dependency parse \mathbf{y} for input sentence \mathbf{x} including words and their parts of speech (POS). Inspired by the parsing model that implements maximum spanning tree (MST) algorithm to induce the dependency parsing tree (McDonald, et al., 2005), the system employs the same framework. The incorporated features are defined over parts of speech of words occurring between and around a possible head-dependent relation.

Suppose $G = (V, E)$ is a directed graph, where V is the set of vertices denoting the words in sentence \mathbf{x} and E is the set of directed edges between any two vertices with some scores. The MST algorithm is to find the most probable subgraph of G that satisfies tree constraints over all vertices. The score function of the parsing tree \mathbf{y} is defined as

$$s(\mathbf{y}) = \sum_{(i,j) \in \mathbf{y}} s(i, j) \quad (1)$$

where $(i, j) \in \mathbf{y}$ indicates an edge in \mathbf{y} from word i to word j and $s(i, j)$ denotes its score. Suppose Y

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

¹<http://homepages.inf.ed.ac.uk/s0450736/maxent.html>

w_i	w_j
p_i	p_j
(w_i, p_i)	(w_j, p_j)
(w_i, w_j)	(p_i, p_j)
(w_i, p_j)	(w_j, p_i)
(w_i, w_j, p_i)	(w_i, w_j, p_j)
(p_i, p_j, w_i)	(p_i, p_j, w_j)
(w_i, w_j, p_i, p_j)	$(p_i, p_k, p_j), i < k < j$
$(p_i, p_{i+1}, p_{j-1}, p_j)$	$(p_{i-1}, p_i, p_{j-1}, p_j)$
$(p_i, p_{i+1}, p_j, p_{j+1})$	$(p_{i-1}, p_i, p_j, p_{j+1})$

Table 1: Features for syntactic parsing.

is the set of syntactic dependency labels, the score function of edges is defined as

$$s(i, j) = \max_{l \in Y} Pr(l|\mathbf{x}, i, j) \quad (2)$$

ME models are used to calculate the value of $Pr(l|\mathbf{x}, i, j)$, where the features are extracted from input sentence \mathbf{x} . Given i and j as the subscripts of words in the sentence and word i is the parent of word j , the features can be illustrated in table 1. w_i and p_i are denoted as the i th word and the i th part of speech respectively in the sentence. The tuples define integrated features, such as (w_i, p_i) indicates the feature combining the i th word and i th part of speech. Besides these features, the distant between word i and word j in sentence \mathbf{x} is considered as a single feature. The distant is also combined with features in table 1 to produce complex features.

2.2 Semantic Dependency Labeling

Semantic dependencies are always concerning with specific predicates. Unlike syntactic dependencies, semantic dependency relationships usually can not be represented as a tree. Thus, the method used for semantic dependency labeling is somewhat different from syntactic dependency parsing. The work of semantic labeling can be divided into two stages: predicate tagging and dependents recognizing.

2.2.1 Predicate Tagging

According to PropBank (Palmer, et al., 2005) and NomBank (Meyers, et al., 2004), predicates usually have several rolesets corresponding to different meanings. For example, the verb *abandon* has three rolesets marked as ordinal numbers **01**, **02** and **03** as described below.

w_i	p_i
p_{i-1}	p_{i+1}
(p_{i-1}, p_i)	(p_i, p_{i+1})
(p_{i-2}, p_i)	(p_i, p_{i+2})
(p_{i-3}, p_i)	(p_i, p_{i+3})
(p_{i-1}, p_i, p_{i+1})	(w_i, p_i)
(w_i, p_{i-1}, p_i)	(w_i, p_i, p_{i+1})
(w_i, p_{i-2}, p_i)	(w_i, p_i, p_{i+2})
(w_i, p_{i-3}, p_i)	(w_i, p_i, p_{i+3})
$(w_i, p_{i-1}, p_i, p_{i+1})$	

Table 2: Features used for predicate tagging.

```

<frameset>
<predicate lemma="abandon">
<roleset id="abandon.01" name="leave
behind" vncls="51.2">
...
</roleset>
<roleset id="abandon.02"
name="exchange" vncls="51.2">
...
</roleset>
<roleset id="abandon.03"
name="surrender, give_over" vncls="-
">
...
</roleset>
</predicate>
</frameset>

```

The goal of this part is to identify the predicates in the sentences and to determine the roleset for each of them. It should be cleared that the ordinal numbers are only used to distinguish different meanings of a predicate. However, if these numbers are treated as tags for predicates, some statistical properties will be obtained as illustrated in Figure 1. As can be seen, the distribution of the *train* data would be quite informative for representing the distribution of other three data sets. Based on this idea, a classification framework is introduced for predicate tagging.

Suppose the tag set is chosen to be $T = \{01, 02, \dots, 22\}$ according to the horizontal axis of Figure 1 and **00** is added to indicate that the examining word is not a predicate. Suppose t_i is a variable indicating the tag of word at position i in sentences \mathbf{x} . ME models are implemented to tag the predicates.

$$t_i = \operatorname{argmax}_{t \in T} Pr(t|\mathbf{x}, i) \quad (3)$$

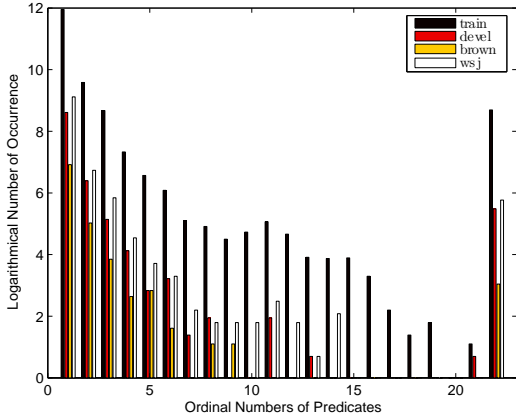


Figure 1: Distribution of the ordinal numbers of predicates on different data sets. 01 - 21 are attached with the predicates in the corpus and 22 stands for ‘SU’.

The features for predicate tagging are listed in table 2, where the symbols share the same meaning as in table 1. Experiments show that this pure statistic processing method is effective for predicate tagging.

2.2.2 Dependents Recognizing

This subtask depends deeply on the results of syntactic parsing and predicate tagging described earlier in the system. Predicate tagging identifies central words and syntactic parsing provides syntactic features for its dependents identification and classification.

Generally speaking, given a specific predicate in a sentence, only a few of words are associated as its semantic dependents. By statistical analysis a list of part of speech tuples that are appearing to be semantic dependency are collected. All other tuples are filtered out to improve system performance.

Suppose (p, d) is a couple of predicate and one of its possible dependents, T is the dependency tree generated by syntactic parsing, L is the set of semantic dependency labels. The dependents can be recognized by using a classification model, ME models are chosen as before.

$$l_{(p,d)} = \operatorname{argmax}_{l \in L} Pr(l|p, d, T) \quad (4)$$

Besides the semantic dependency labels, *null* is included as a special tag to indicate that there is no semantic dependency between p and d . As a result, dependents identification (binary classification) and dependents tagging (multi-classification) can

be solved together within one multi-classification framework.

The selected features are listed below.

1. Predicate Features

- **Lemma and POS** of predicate, predicate’s parent in syntactic dependency tree.
- **Voice** active or passive.
- **Syntactic dependency label** of edge between predicate and its parent.
- **POS framework** POS list of predicate’s siblings, POS list of predicate’s children.
- **Syntactic dependency framework** syntactic dependency label list of the edges between predicate’s parent and its siblings.
- **Parent framework** syntactic dependency label list of edges connecting to predicate’s parent.

2. Dependent Features

- **Lemma and POS** of dependent, dependent’s parent.
- **POS framework** POS list of dependent’s siblings.
- **Number of children** of dependent’s parent.

3. In Between Features

- **Position** of dependent according to predicate: before or after.
- **POS pair** of predicate and dependent.
- **Family relation** between predicate and dependent: ancestor or descendant.
- **Path length** between predicate and dependent.
- **Path POS** POS list of all words appearing on the path from predicate to dependent.
- **Path syntactic dependency label** list of dependency label of edges of path between predicate and dependent.

3 Experiment results

The classification models were trained using all the training data. The detailed information are shown in table 3. All experiments ran on 32-bit Intel(R) Pentium(R) D CPU 3.00GHz processors with 2.0G memory.

	Feature Number	Training Time
<i>Syn.</i>	7,488,533	30h
<i>Prd.</i>	1,484,398	8h
<i>Sem.</i>	3,588,514	12h

Table 3: Details of ME models. *Syn.* is for syntactic parsing, *Prd.* is for predicate tagging and *Sem.* is for semantic dependents recognizing.

	Syntactic	Semantic	Overall
devel	85.29	69.60	77.49
brown	80.80	59.17	70.01
wsj	87.42	71.27	79.38
brown+wsj	86.69	69.95	78.35

(a) Closed Challenge

	Syntactic	Semantic	Overall
devel	85.29	68.45	76.87
brown	80.80	58.22	69.51
wsj	87.42	70.32	78.87
brown+wsj	86.69	68.99	77.84

(b) Open Challenge

Table 4: Scores for joint learning of syntactic and semantic dependencies.

3.1 Closed Challenge

The system for closed challenge is designed as a two-stage parser: syntactic parsing and semantic dependency labeling as described previously. Table 4(a) shows the results on different corpus. As shown in table 4(a), the scores of semantic dependency labeling are quite low, that are influencing the overall scores. The reason could be inferred from the description in section 2.2.2 since semantic dependent labeling inherits the errors from the output of syntactic parsing and predicate tagging. Following evaluates each part independently.

Besides the multiple classification model described in table 3, a binary classification model was built based on ME for predicate tagging. The binary model can't distinguish different rolesets of predicate, but can identify which words are predicates in sentences. The precision and recall for binary model are 90.80 and 88.87 respectively, while for multiple model, the values are 84.60 and 85.60.

For semantic dependent labeling, experiments were performed under conditions that the gold syntactic dependency tree and predicates list were given as input. The semantic scores became 80.09, 77.08 and 82.25 for *devel*, *brown* and *wsj* respectively. This implies that the error of syntactic pars-

ing and predicate tagging could be probably augmented in semantic dependent labeling. In order to improve the performance of the whole system, the deep dependence between the two stages should be broken up in future research.

3.2 Open Challenge

In open challenge, the same models are used for syntactic parsing and predicate tagging as in closed challenge and two other models are trained for semantic dependent labeling. Suppose M_{mst} , M_{malt} and M_{chunk} are denoted as these three semantic models, where M_{mst} is the model used in closed challenge, M_{malt} is trained on the syntactic dependency tree provided by the open corpus with the same feature set as M_{mst} , and M_{chunk} is trained using features extracted from name entity and wordnet super senses results provided by the open corpus.

Considering a possible dependent given a specific predicate, the feature set used for M_{chunk} contains only six elements:

- Whether the dependent is in name entity chunk: True or False.
- Name entity label of the dependent.
- Whether the dependent is in BBN name entity chunk: True or False.
- BBN name entity label of the dependent.
- Whether the dependent is in wordnet super sense chunk: True or False.
- Wordnet super sense label of the dependent.

After implementing these three models on semantic dependents recognizing, the results were merged to generate the scores described in table 4(b).

The merging strategy is quite simple. Given a couple of predicate and dependent (p, d) , the system produces three semantic dependency labels denoting as l_{mst} , l_{malt} and l_{chunk} , the result label is chosen to be most frequent semantic label among the three.

Comparing the scores of open challenge and closed challenge, it can be found that the score of the former is less than the latter, which is quite strange since more resources were used in open challenge. To examine the influences of different semantic dependents recognizing models, each

	M_{mst}	M_{malt}	M_{chunk}
devel	69.60	64.48	41.72
brown	59.17	56.52	34.04
wsj	71.27	66.40	41.83

Table 5: Semantic scores of different models.

model was implemented in the closed challenge and the results are shown in table 5. Specially, model M_{chunk} generated too low scores and gave a heavy negative influence on the final results. Finding a good way to combine several results requires further research.

4 Conclusions

This paper have presented a simple discriminative system submitted to the CoNLL-2008 shared task to address the learning task of syntactic and semantic dependencies. The system was divided into syntactic parsing and semantic dependents labeling. Maximum spanning tree was used to find a syntactic dependency tree in the full-connected graph constructed over the words of a sentence. Maximum entropy models were implemented to classify syntactic dependency edges, predicates and their semantic dependents. A brief analysis has also been given on the results of both closed challenge and open challenge.

Acknowledgement

This research has been partially supported by the National Natural Science Foundation of China (No. 60435020 and No. 90612005), the Goal-oriented Lessons from the National 863 Program of China (No.2006AA01Z197) and Project of Microsoft Research Asia. We would like to thank Zhixin Hao, Xiao Xin, Languang He and Tao Qian for their wise suggestion and great help. Thanks also to Muhammad Waqas Anwar for English improvement.

References

- Adam Berger, Stephen Della Pietra, Vincent Della Pietra 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39-71.
- Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young and Ralph Grishman 2004. The NomBank Project: An Interim Report *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, 24-31.

Martha Palmer, Daniel Gildea, Paul Kingsbury 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles *Computational Linguistics*, 31(1):71-106.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez and Joakim Nivre 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič 2005. Non-projective Dependency Parsing using Spanning Tree Algorithms. *Proceedings of HLT/EMNLP*, 523-530.

Discriminative vs. Generative Approaches in Semantic Role Labeling

Deniz Yuret
Koç University
dyuret@ku.edu.tr

Mehmet Ali Yatbaz
Koç University
myatbaz@ku.edu.tr

Ahmet Engin Ural
Koç University
aural@ku.edu.tr

Abstract

This paper describes the two algorithms we developed for the CoNLL 2008 Shared Task “Joint learning of syntactic and semantic dependencies”. Both algorithms start parsing the sentence using the same syntactic parser. The first algorithm uses machine learning methods to identify the semantic dependencies in four stages: identification and labeling of predicates, identification and labeling of arguments. The second algorithm uses a generative probabilistic model, choosing the semantic dependencies that maximize the probability with respect to the model. A hybrid algorithm combining the best stages of the two algorithms attains 86.62% labeled syntactic attachment accuracy, 73.24% labeled semantic dependency F1 and 79.93% labeled macro F1 score for the combined WSJ and Brown test sets¹.

1 Introduction

In this paper we describe the system we developed for the CoNLL 2008 Shared Task (Surdeanu et al., 2008). Section 2 describes our approach for identifying syntactic dependencies. For semantic role labeling (SRL), we pursued two independent approaches. Section 3 describes our first approach, where we treated predicate identification and labeling, and argument identification and labeling as

four separate machine learning problems. The final program consists of four stages, each stage taking the answers from the previous stage as given and performing its own identification or labeling task based on a model generated from the training set. Section 4 describes our second approach where we used a generative model based on the joint distribution of the predicate, the arguments, their labels and the syntactic dependencies connecting them. Section 5 summarizes our results and suggests possible improvements.

2 Syntactic dependencies

We used a non-projective dependency parser based on spanning tree algorithms. The parameters were determined based on the experimental results of the English task in (McDonald et al., 2005), i.e. we used projective parsing and a first order feature set during training. Due to the new representation of hyphenated words in both training and testing data of our shared task and the absence of the gold part of speech (GPOS) column in the test data, the format of the CoNLL08 shared task is slightly different from the format of the CoNLL05 shared task, which is supported by the McDonald’s parser. We reformatted the data accordingly. The resulting labeled attachment score on the test set is 87.39% for WSJ and 80.46% for Brown.

3 The 4-stage discriminative approach

Our first approach to SRL consists of four distinct stages: (1) predicate identification, (2) predicate labeling, (3) argument identification, and (4) argument labeling.

A discriminative machine learning algorithm is trained for each stage using the gold input and output values from the training set. The following sec-

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

¹These numbers are slightly higher than the official results due to a small bug in our submission.

tions describe the machine learning algorithm, the nature of its input/output, and the feature selection process for each stage. The performance of each stage is compared to a most frequent class baseline and analyzed separately for the two test sets and for nouns and verbs. In addition we look at the performance given the input from the gold data vs. the input from the previous stage.

3.1 Predicate identification

The task of this stage is to determine whether a given word is a nominal or a verb predicate using the dependency-parsed input. As potential predicates we only consider words that appear as a predicate in the training data or have a corresponding PropBank or NomBank XML file. The method constructs feature vectors for each occurrence of a target word in the training and test data. It assigns class labels to the target words in the training data depending on whether a target word is a predicate or not, and finally classifies the test data. We experimented with combinations of the following features for each word in a $2k + 1$ word window around the target: (1) POS(W): the part of speech of the word, (2) DEP(W, HEAD(W)): the syntactic dependency of the word, (3) LEMMA(W): the lemma of the word, (4) POS(HEAD(W)): the part of speech of the syntactic head.

We empirically selected the combination that gives the highest accuracy in terms of the precision and recall scores on the development data. The method achieved its highest score when we used features 1-3 for the target word and features 1-2 for the neighbors in a $[-3 +3]$ word window. TiMBL (Daelemans et al., 2004) was used as the learning algorithm.

Table 1 (4-stage, All1) shows the results of our learning method on the WSJ and Brown test data. The noun and verb results are given separately (Verb1, Noun1). To distinguish the mistakes coming from parsing we also give the results of our method after the gold parse (4-stage-gold). Our results are significantly above the most frequent class baseline which gives 72.3% on WSJ and 65.3% on Brown.

3.2 Predicate labeling

The task of the second stage is deciding the correct frame for a word given that the word is a predicate. The input of the stage is 11-column data, where the columns contain part of speech, lemma and syntactic dependency for each word. The first stage's

decision for the frame is indicated by a string in the predicate column. The output of the stage is simply the replacement of that string with the chosen frame of the word. The chosen frame of the word may be word.X, where X is a valid number in PropBank or NomBank.

The statistics of the training data show that by picking the most frequent frame, the system can pick the correct frame in a large percent of the cases. Thus we decided to use the most frequent frame baseline for this stage. If the word is never seen in the training, first frame of the word is picked as default.

In the test phase, the results are as the following; in the Brown data, assuming that the stage 1 is gold, the score is 80.8%, noting that 11% of the predicates are not seen in the training phase. In WSJ, the score based on gold input is 88.3%, and only 5% of the predicates are not seen in the training phase. Table 1 gives the full results for Stage 2 (4-stage, Verb2, Noun2, All2).

3.3 Argument identification

The input data at this stage contains the syntactic dependencies, predicates and their frames. We look at the whole sentence for each predicate and decide whether each word should be an argument of that predicate or not. We mark the words we choose as arguments indicating which predicate they belong to and leave the labeling of the argument type to the next stage. Thus, for each predicate-word pair we have a yes/no decision to make.

As input to the learning algorithm we experimented with representations of the syntactic dependency chain between the predicate and the argument at various levels of granularity. We identified the syntactic dependency chain between the predicate and each potential argument using breadth-first-search on the dependency tree. We tried to represent the chain using various subsets of the following elements: the argument lemma and part-of-speech, the predicate frame and part-of-speech, the parts-of-speech and syntactic dependencies of the intermediate words linking the argument to the predicate.

The syntactic dependencies leading from the argument to the predicate can be in the head-modifier or the modifier-head direction. We marked the direction associated with each dependency relation in the chain description. We also experimented

with using fine-grained and coarse-grained parts of speech. The coarse-grained part of speech consists of the first two characters of the Penn Treebank part of speech given in the training set.

We used a simple learning algorithm: choose the answer that is correct for the majority of the instances with the same chain description from the training set. Not having enough detail in the chain description leaves crucial information out that would help with the decision process, whereas having too much detail results in bad classifications due to sparse data. In the end, neither the argument lemma, nor the predicate frame improved the performance. The best results were achieved with a chain description including the coarse parts of speech and syntactic dependencies of each word leading from the argument to the predicate. The results are summarized in Table 1 (4-stage, Verb3, Noun3, All3).

3.4 Argument labeling

The task of this stage is choosing the correct argument tag for a modifier given that it is modifying a particular predicate. Input data format has additional columns indicating which words are arguments for which predicates. There are 54 possible values for a labeled argument. As a baseline we take the most frequent argument label in the training data (All1) which gives 37.8% on the WSJ test set and 33.8% on the Brown test set.

The features to determine the correct label of an argument are either lexical or syntactic. In a few cases, they are combined. The following list gives the set we have used. Link is the type of the syntactic dependency. Direction is left or right, depending the location of the head and the modifier in the sentence. LastLink is the type of the dependency at the end of the dependency chain and firstLink is type of the dependency at the beginning of the dependency chain.

Feature1 : modifierStem + headStem

Feature2 : modifierStem + coarsePosModifier + headStem + coarsePosHead + direction

Feature3 : coarsePosModifier + headPos + firstLink + lastLink + direction

Feature4: modifierStem + coarsePosModifier

The training phase includes building simple histograms based on four features. Feature1 and Feature2 are sparser than the other two features and are better features as they include lexical information. Last two features are less sparse, covering

most of the development data, i.e. their histograms give non-zero values in the development phase. In order to match all the instances in the development and use the semantic information, a cascade of the features is implemented similar to the one done by Gildea and Jurafsky(2002), although no weighting and a kind of back-off smoothing is used. First, a match is searched in the histogram of the first feature, if not found it is searched in the following histogram. After a match, the most frequent argument with that match is returned. Table 1 gives the performance (4-stage, Verb4, Noun4, All4).

4 The generative approach

One problem with the four-stage approach is that the later stages provide no feedback to the earlier ones. Thus, a frame chosen because of its high prior probability will not get corrected when we fail to find appropriate arguments for it. A generative model, on the other hand, does not suffer from this problem. The probability of the whole assignment, including predicates, arguments, and their labels, is evaluated together and the highest probability combination is chosen.

4.1 The generative model

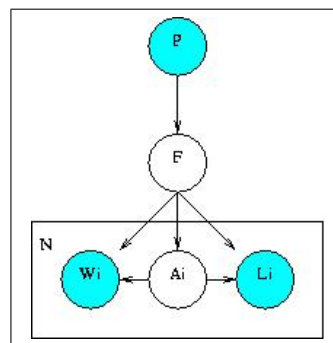


Figure 1: The graphical model depicting the conditional independence assumptions.

Our generative model specifies the distribution of the following random variables: P is the lemma (stem+pos) of a candidate predicate. F is the frame chosen for the predicate (could be null). A_i is the argument label of word i with respect to a given predicate (could be null). W_i is the lemma (stem+pos) of word i . L_i is the syntactic dependency chain leading from word i to the given predicate (similar to Section 3.3).

We consider each word in the sentence as a candidate predicate and use the joint distribution of the above variables to find the maximum probability F

WSJ	Verb1	Verb2	Verb3	Verb4	Noun1	Noun2	Noun3	Noun4	All1	All2	All3	All4
4-stage	97.1	85.5	85.7	71.7	84.6	78.4	61.1	49.4	90.6	81.8	76.6	63.5
generative	96.1	88.4	83.4	74.0	82.8	79.5	69.8	63.2	89.0	83.6	77.4	69.2
4-stage-gold	97.4	88.3	95.2	82.7	85.2	92.7	70.5	81.9	91.1	90.5	86.0	82.4
generative-gold	96.3	92.6	91.1	88.0	83.4	95.5	80.7	86.9	89.4	94.0	86.7	87.5
hybrid	97.1	89.3	85.7	74.7	84.6	80.9	70.9	64.0	90.6	84.9	79.5	70.2

Brown	Verb1	Verb2	Verb3	Verb4	Noun1	Noun2	Noun3	Noun4	All1	All2	All3	All4
4-stage	93.0	74.5	78.9	59.0	74.4	58.6	52.3	38.8	86.0	68.6	72.8	54.3
generative	91.4	71.7	76.1	60.0	70.8	59.3	54.0	45.3	83.1	66.6	69.6	55.7
4-stage-gold	93.0	80.8	93.7	73.2	75.7	80.3	70.1	70.5	86.5	80.8	88.2	72.4
generative-gold	91.6	80.6	85.8	78.05	71.2	85.9	70.5	75.1	83.5	82.6	81.8	77.1
hybrid	93.0	73.3	78.9	60.4	74.4	62.9	57.6	47.5	86.0	69.3	73.4	57.0

Table 1: The F1 scores for different datasets, models, stages, and predicate parts of speech. The ‘‘Verb’’ in the column heading indicates verbal predicates, ‘‘Noun’’ indicates nominal predicates, ‘‘All’’ indicates all predicates. The numbers 1-4 in column headings indicate the 4 stages: (1) predicate identification, (2) predicate labeling, (3) argument identification, (4) argument labeling. The gold results assume perfect output from the previous stages. The highest number in each column is marked with boldface.

and A_i labels given P , W_i , and L_i . The graphical model in Figure 1 specifies the conditional independence assumptions we make. Equivalently, we take the following to be proportional to the joint probability of a particular assignment:

$$\Pr(F|P) \prod_i \Pr(A_i|F) \Pr(W_i|FA_i) \Pr(L_i|FA_i)$$

4.2 Parameter estimation

To estimate the parameters of the generative model we used the following methodology:

For $\Pr(F|P)$ we use the maximum likelihood estimate from the training data. As a consequence, frames that were never observed in the training data have zero probability. One exception is lemmas which have not been observed in the training data, for which each frame is considered equally likely.

For $\Pr(A_i|F)$ we also use the maximum likelihood estimate and normalize it using sentence length. For a given argument label we find the expected number of words in a sentence with that label for frame F . We divide this expected number with the length of the given sentence to find $\Pr(A_i|F)$ for a single word. Any leftover probability is given to the null label. If the sentence length is shorter than the expected number of arguments, all probabilities are scaled down proportionally.

For the remaining two terms $\Pr(L_i|F, A_i)$ and $\Pr(W_i|F, A_i)$ using the maximum likelihood estimate is not effective because of data sparseness. The arguments in the million word training data contain about 16,000 unique words and 25,000

unique dependency chains. To handle the sparseness problem we smoothed these two estimates using the part-of-speech argument distribution, i.e. $\Pr(L_i|\text{POS}, A_i)$ and $\Pr(W_i|\text{POS}, A_i)$, where POS represents the coarse part of speech of the predicate.

5 Results and Analysis

Table 1 gives the F1 scores for the two models (4-stage and generative), presented separately for noun and verb predicates and the four stages of predicate identification/labeling, argument identification/labeling. In order to isolate the performance of each stage we also give their scores with gold input. The rest of this section analyzes these results and suggests possible improvements.

A hybrid algorithm: A comparison of the two algorithms show that the 4-stage approach is superior in predicate and verbal-argument identification and the generative algorithm is superior in the labeling of predicates and arguments and nominal-argument identification. This suggests a hybrid algorithm where we restrict the generative model to take the answers for the better stages from the 4-stage algorithm (Noun1, Verb1, Verb3) as given. Tables 1 and 2 present the results for the hybrid algorithm compared to the 4-stage and generative models.

Parsing performance: In order to see the effect of syntactic parsing performance, we ran the hybrid algorithm starting with the gold parse. The labeled semantic score went up to 78.84 for WSJ and 67.20 for Brown, showing that better parsing

Data/algorithm	Unlabeled	Labeled
WSJ 4-stage	81.15	69.44
WSJ generative	81.01	73.66
WSJ hybrid	82.94	74.74
Brown 4-stage	76.91	58.76
Brown generative	73.76	59.05
Brown hybrid	77.22	60.80

Table 2: Semantic scores for the 4-stage, generative, and hybrid algorithms

can add about 4-6% to the overall performance.

Syntactic vs lexical features: Our algorithms use two broad classes of features: information from the dependency parse provides syntactic evidence, and the word pairs themselves provide semantic evidence for a possible relation. To identify their relative contributions, we experimented with two modifications of the generative algorithm: *gen-l* does not use the $\Pr(W_i|FA_i)$ term and *gen-w* does not use the $\Pr(L_i|FA_i)$ term. *gen-l*, using only syntactic information and the predicate, gets a labeled semantic score of 70.97 for WSJ and 58.83 for Brown, a relatively small decrease. In contrast *gen-w*, using only lexical information gets 43.06 for WSJ and 33.17 for Brown causing almost a 40% decrease in performance.

On the other hand, we find that the lexical features are essential for certain tasks. In labeling the arguments of nominal predicates, finding an exact match for the lexical pair guarantees a 90% accuracy. If there is no exact match, the 4-stage algorithm falls back on a syntactic match, which only gives a 75% accuracy.

Future work: The hybrid algorithm shows the strengths and weaknesses of our two approaches. The generative algorithm allows feedback from the later stages to the earlier stages and the 4-stage machine learning approach allows the use of better features. One way to improve the system could be by adding feedback to the 4-stage algorithm (later stages can veto input coming from previous ones), or adding more features to the generative model (e.g. information about neighbor words when predicting F). More importantly, there is no feedback between the syntactic parser and the semantic role labeling in our systems. Treating both problems under the same framework may lead to better results.

Another property of both models is the indepen-

dence of the argument label assignments from each other. Even though we try to control the number of arguments of a particular type by adjusting the parameters, there are cases when we end up with no assignments for a mandatory argument or multiple assignments where only one is allowed. A more strict enforcement of valence constraints needs to be studied.

The use of smoothing in the generative model was critical, it added about 20% to our final F1 score. This raises the question of finding more effective smoothing techniques. In particular, the jump from specific frames to coarse parts of speech is probably not optimal. There may be intermediate groups of noun and verb predicates which share similar semantic or syntactic argument distributions. Identifying and using such groups will be considered in future work.

References

- Daelemans, W., J. Zavrel, K. van der Sloot, and A. van den Bosch. 2004. TiMBL: Tilburg memory-Based Learner. Tilburg University.
- Gildea, D. and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245-288.
- McDonald, R., K. Crammer, and F. Pereira. 2005. Online Large-Margin Training of Dependency Parsers. *Ann Arbor*, 100.
- Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.

A Pipeline Approach for Syntactic and Semantic Dependency Parsing

Yotaro Watanabe and Masakazu Iwatate and Masayuki Asahara and Yuji Matsumoto

Nara Institute of Science and Technology, Japan

8916-5, Takayama, Ikoma, Nara, Japan, 630-0192

{yotaro-w, masakazu-i, masayu-a, matsu}@is.naist.jp

Abstract

This paper describes our system for syntactic and semantic dependency parsing to participate the shared task of CoNLL-2008. We use a pipeline approach, in which syntactic dependency parsing, word sense disambiguation, and semantic role labeling are performed separately: Syntactic dependency parsing is performed by a tournament model with a support vector machine; word sense disambiguation is performed by a nearest neighbour method in a compressed feature space by probabilistic latent semantic indexing; and semantic role labeling is performed by an online passive-aggressive algorithm. The submitted result was 79.10 macro-average F1 for the joint task, 87.18% syntactic dependencies LAS, and 70.84 semantic dependencies F1. After the deadline, we constructed the other configuration, which achieved 80.89 F1 for the joint task, and 74.53 semantic dependencies F1. The result shows that the configuration of pipeline is a crucial issue in the task.

1 Introduction

This paper presents the description of our system in CoNLL-2008 shared task. We split the shared task into five sub-problems – syntactic dependency parsing, syntactic dependency label classification, predicate identification, word sense disambiguation, and semantic role labeling. The overview of our system is illustrated in Figure 1. Our de-

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

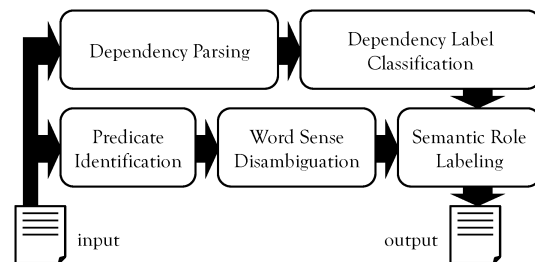


Figure 1: Overview of the System

pendency parsing module is based on a tournament model (Iida et al., 2003), in which a dependency attachment is estimated in step-ladder tournament matches. The relative preference of the attachment is modeled by one-on-one match in the tournament. Iwatate et al. (Iwatate et al., 2008) initially proposed the method for Japanese dependency parsing, and we applied it to other languages by relaxing some constraints (Section 2.1). Dependency label classification is performed by a linear-chain sequential labeling on the dependency siblings like McDonald’s schemata (McDonald et al., 2006). We use an online passive-aggressive algorithm (Crammer et al., 2006) for linear-chain sequential labeling (Section 2.2). We also use the other linear-chain sequential labeling method to annotate whether each word is a predicate or not (Section 2.3). If an identified predicate has more than one sense, a nearest neighbour classifier disambiguates the word sense candidates (Section 2.4). We use an online passive-aggressive algorithm again for the semantic role labeling (Section 2.5). The machine learning algorithms used in separated modules are diverse due to role sharing.¹

¹Unlabeled dependency parsing was done by Iwatate, dependency label classification and semantic role labeling was done by Watanabe, predicate identification and word sense

We attempt to construct a framework in which each module passes k-best solutions and the last semantic role labeling module performs reranking of the k-best solutions using the overall information. Unfortunately, we couldn't complete the framework before the deadline of the test run. Our method is not a "joint learning" approach but a pipeline approach.

2 Methods

2.1 Unlabeled Dependency Parsing

The detailed description of the tournament model-based Japanese dependency parsing is found in (Iwatate et al., 2008). The original Iwatate's parsing algorithm was for Japanese, which is for a strictly head-final language. We adapt the algorithm to English in this shared task. The tournament model chooses the most likely candidate head of each of the focused words in a step-ladder tournament. For a given word, the algorithm repeats to compare two candidate heads and finds the most plausible head in the series of a tournament. On each comparison, the winner is chosen by an SVM binary classifier with a quadratic polynomial kernel². The model uses different algorithms for training example generation and parsing. Figures 2 and 3 show training example generation and parsing algorithm, respectively. Time complexity of both algorithms is $O(n^2)$ for the number of words in an input sentence. Below, we present the features for SVM

```
// N: # of tokens in input sentence
// true_head[j]: token j's head at
//           training data
// gen(j,i1,i2,LEFT): generate an example
//   where token j is dependent of i1
// gen(j,i1,i2,RIGHT): generate an example
//   where token j is dependent of i2
// Token 0 is the virtual ROOT.

for j = 1 to N-1 do
  h = true_head[j];
  for i = 0 to h-1 do
    if i!=j then gen(j,i,h,RIGHT);

    for i = h+1 to N do
      if i!=j then gen(j,h,i,LEFT);
end-for;
```

Figure 2: Pseudo Code of Training Example Generation

disambiguation was done by Asahara, and all tasks were supervised by Matsumoto.

²We use TinySVM as an SVM classifier. chasen.org/~taku/software/TinySVM/

```
// N: # of tokens in input sentence
// head[]: (analyzed-) head of tokens
// classify(j,i1,i2): ask SVM
//   which candidate (i1 or i2) is
//   more likely for head of j.
//   return LEFT if i1 wins.
//   return RIGHT if i2 wins.
// cand.push_back(k): add token index k
//   to the end of cand.
// cand.erase(i): remove i-th element
//   from cand.

for j = 1 to N do
  cand = [];
  for i = 0 to N do
    if i!=j then cand.push_back(i);
  end-for;

  while cand.size() > 1 do
    if classify(j,cand[0],
              cand[1]) = LEFT then
      cand.erase(1);
    else
      cand.erase(0);
    end-if;
  end-while;

  head[j] = cand[0];
end-for;
```

Figure 3: Pseudo Code of Parsing Algorithm

in our tournament model. The FORM, LEMMA, GPOS(for training), PPOS(for testing, instead of GPOS), SPLIT_FORM, SPLIT_LEMMA, PPOSS in the following tokens were used as the features:

- Dependent, candidate1, candidate2
- Immediately-adjacent tokens of dependent, candidate1, candidate2, respectively
- All tokens between dependent-candidate1, dependent-candidate2, candidate1-candidate2, respectively

We also used the distance feature: distance (1 or 2-5 or 6+ tokens) between dependent-candidate1, dependent-candidate2, and candidate1-candidate2. Features corresponding to the candidates, including the distance feature, have a prefix that indicates its side: "L-"(the candidate appears on left-hand-side of the dependent) or "R-"(appears on right-hand-side of the dependent). Training an SVM model with all examples is time-consuming, and split the examples by the dependent GPOS for training (PPOS for testing, instead of GPOS³) to run SVM training in parallel. Since the number of examples with the dependent PPOS:IN, NN, NNP

³We cannot use GPOS for testing due to the shared task regulation.

is still large, we used only first 1.5 million examples for the dependent GPOS. Note that, the algorithm does not check the well-formedness of dependency trees⁴.

2.2 Dependency Label Classification

This phase labels a dependency relation label to each word in a parse tree produced in the preceding phase. (McDonald et al., 2006) suggests that edges of head x_i and its dependents x_{j1}, \dots, x_{jM} are highly correlated, and capturing these correlation improves classification accuracy. In their approach, edges of a head and its dependents $e_{i,j1}, \dots, e_{i,jM}$ are classified sequentially, and then Viterbi algorithm is performed to find the highest scoring label sequence. We take a similar approach with some simplification. In our system, each edge is classified deterministically, and the previous decision is used as a feature for the subsequent classification.

We use an online passive aggressive algorithm (Crammer et al., 2006)⁵ for dependency label classification since it converges fast, gives good performance and can be implemented easily. The features used in this phase are primarily similar to that of (McDonald et al., 2006).

Word features: SPLIT_LEMMA, PPOS, affix (lengths 2 and 3) of the head and the dependent.

Position: Position relation between the head and the dependent (Is the head anterior to dependent?). Is the word top of the sentence? Is the word last of the sentence?

Context features: SPLIT_LEMMA, PPOS, affix (lengths 2 and 3) of the nearest left/right word. SPLIT_LEMMA and PPOS bigram (ww, wp, pw, pp) of the head and the dependent (window size 5).

Sibling features: SPLIT_LEMMA, PPOS, affix (lengths 2 and 3) of the dependent’s nearest left and right siblings in the dependency tree.

Other features: The number of dependent’s children. Whether the dependent and the dependent’s grand parent SPLIT_LEMMA/PPOS are the same. The previous classification result (previous label).

2.3 Predicate Identification

This phase solves which word can be a predicate. In the predicate spotting, the linear-chain

⁴We tried to make a k-best cascaded model among the modules. The latter module can check the well-formedness of the tree. The current implementation skips this well-formedness checking.

⁵We use PA algorithm among PA, PA-I and PA-II in (Crammer et al., 2006).

CRF (Lafferty et al., 2001) annotates whether the word is a predicate or not. The FORM, LEMMA (itself, and whether the LEMMA is registered in the PropBank/NomBank frames), SPLIT_FORM, SPLIT_LEMMA, PPOSS within 5 token window size are used as the features. We also use bigram features within 3 token window size and trigram features within 5 token window size for FORM, LEMMA, SPLIT_FORM, SPLIT_LEMMA, and PPOSS. The main reason why we use a sequence labeling method for predicate identification was to relax the effect of the tagging error of PPOS and PPOSS. However, we will show later that this module aggravates the total performance.

2.4 Word Sense Disambiguation

For the word sense disambiguation, we use 1-nearest neighbour method in a compressed feature space by probabilistic latent semantic indexing (PLSI). We trained the word sense disambiguation model from the example sentences in the training/development data and PropBank/NomBank frames. The metric in the nearest neighbour method is based on the occurrence of LEMMA in the example sentences. However, the examples in the PropBank/NomBank do not contain the lemma information. To lemmatize the words in the PropBank/NomBank, we compose a lemmatizer from the FORM-LEMMA table in the training and development.⁶ Since the metric space is very sparse, PLSI (Hofmann, 1999) is used to reduce the metric space dimensions. We used KL-divergence between two examples of $P(d_i|z_k)$ of $P(d_i, w_j) = \sum_k P(d_i|z_k)P(w_j|z_k)P(z_k)$ as hemi-metric for the nearest neighbour method⁷, in which $d_i \in D$ is an example sentence in the training/devel/test data and PropBank/NomBank frames; $w_j \in W$ is LEMMA; and $z_k \in Z$ is a latent class. We use $|Z| = 100$, which gave the best performance in the development data. Note, we transductively used the test data for the PLSI modeling within the test run period.

2.5 Semantic Role Labeling

While semantic role labeling task is generally performed by two phases: argument identification and argument classification, we did not divide the task

⁶We are not violating the closed track regulation to build the lemmatizer. If a word in the PropBank/NomBank is not in the training/development data, we give up lemmatization.

⁷We use $D_{KL} = \sum_k P(d_{input\ data}|z_k) \log \frac{P(d_{input\ data}|z_k)}{P(d_{1-nearest\ data}|z_k)}$ as hemi-metric. It is a non-commutative measure.

into the two phases. That is, argument candidates are directly assigned a particular semantic role label. We did not employ any candidate filtering procedure, so argument candidates consist of words in any predicate-word pair. The argument candidates that have no roles assigned are assigned “NONE” label. For the reason that described in Section 2.2 (fast convergence and good performance), we use an on-line passive aggressive algorithm for learning the semantic role classifiers.

Useful features for argument classification of verb and noun predicates are different. For example, voice (active or passive) is essential for verb predicate’s argument classification. On the other hand, presence of a genitive word is useful for noun predicate’s argument classification. For this reason, we created two models: argument classifier for verb predicates and that for noun predicates.

Semantic frames are useful information for semantic role classification. Generally, obligatory arguments not included in semantic frames do not appear in actual texts. For this reason, we use PropBank/NomBank semantic frames for semantic role pruning. Suppose semantic roles in the semantic frame are $F_i = \{A0, A1, A2, A3\}$. Since obligatory arguments are $\{A0...AA\}$, the remaining arguments $\{A4, A5, AA\}$ are removed from label candidates.

For verb predicates, the features used in our system are based on (Hacioglu, 2004). We also employed some other features proposed in (Gildea and Jurafsky, 2002; Pradhan et al., 2004b). For noun predicates, the features are primarily based on (Pradhan et al., 2004a). The features that we defined for semantic role labeling are as follows:

Word features: SPLIT_LEMMA and PPOS of the predicate, dependent and dependent’s head, and its conjunctions.

Dependency label: The dependency label between the argument candidate and the its head.

Family: The position of the argument candidate with respect to the predicate position over the dependency tree (e.g., child, sibling).

Position: The position of the head of the dependency relation with respect to the predicate position in the sentence.

Pattern: The left-to-right chain of the PPOS/dependency labels of the predicate’s children.

Context features: PPOS of the nearest left/right word.

Path features: SPLIT_LEMMA, PPOS and dependency label paths between the predicate and the argument candidate, and its path bi-gram.

Distance: The number of paths between the predicate and the argument candidate.

Voice: Voice of the predicate (active or passive) and voice-position conjunction (for verb predicates).

Is predicate plural: Whether the predicate is singular or plural (for noun predicates).

Genitives between the predicate and the argument: Is there a genitive word between the predicate and the argument? (for noun predicates)

3 Results

Table 1 shows the result of our system. The proposed method was effective in dependency parsing (rank 3rd), but was not good in semantic role labeling (rank 9th). One reason of the result of semantic role labeling could be usages of PropBank/NomBank frames. We did not achieve the maximum use of the resources, hence the design of features and the choice of learning algorithm may not be optimal.

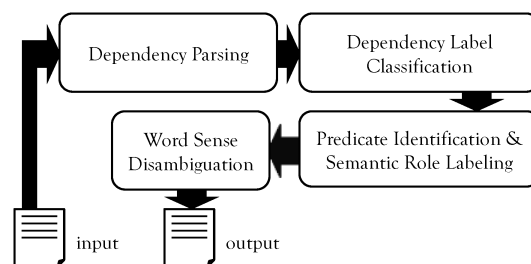


Figure 4: Overview of the Modified System

The other reason is the design of the pipeline. We changed the design of the pipeline after the test run. The overview of the modified system is illustrated in Figure 4. After the syntactic dependency parsing, we limited the predicate candidates as verbs and nouns by PPOSS, and filtered the argument candidates by Xue’s method (Xue and Palmer, 2004). Next, the candidate pair of predicate-argument was classified by an online passive-aggressive algorithm as shown in Section 2.5. Finally, the word sense of the predicate is determined by the module in Section 2.4. The new result is scores with * in Table 1. The result means that the first design was not the best for the task.

Acknowledgements

We would like to thank the CoNLL-2008 shared task organizers and the data providers (Surdeanu et al., 2008).

Problem	All	WSJ	Brown	Rank
Complete Problem	79.10 (80.89*)	80.30 (82.06*)	69.29 (71.32*)	9th
Semantic Dependency	70.84 (74.53*)	72.37 (76.01*)	58.21 (62.41*)	9th
Semantic Role Labeling	67.92 (72.31*)	69.31 (73.62*)	56.42 (61.64*)	-
Predicate Identification & Word Sense Disambiguation	77.20 (79.17*)	79.02 (80.99*)	62.10 (64.03*)	-
Syntactic Dependency (Labeled)	87.18	88.06	80.17	3rd
Syntactic Label Accuracy	91.63	92.31	86.26	-
Unlabeled Syntactic Dependency Unlabeled	90.20	90.73	85.94	-

The scores with * mark are our post-evaluation results.

Table 1: The Results – Closed Challenge

References

- Buchholz, Sabine and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *CoNLL-2006: Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164.
- Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwarz, and Yoram Singer. 2006. Online Passive-Agressive Algorithms. *Journal of Machine Learning Research*, 7:551–585.
- Gildea, Daniel and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.
- Hacioglu, Kadri. 2004. Semantic role labeling using dependency trees. In *COLING-2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1273–1276.
- Hofmann, Thomas. 1999. Probabilistic Latent Semantic Indexing. In *SIGIR-1999: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Informative Retrieval*, pages 50–57.
- Iida, Ryu, Kentaro Inui, Hiroya Takamura, and Yuji Matsumoto. 2003. Incorporating Contextual Cues in Trainable Models for Coreference Resolution. In *EACL Workshop ‘The Computational Treatment of Anaphora’*, pages 23–30.
- Iwatate, Masakazu, Masayuki Asahara, and Yuji Matsumoto. 2008. Japanese Dependency Parsing Using a Tournament Model. In *COLING-2008: Proceedings of the 22nd International Conference on Computational Linguistics (To Appear)*.
- Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML-2001: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.
- McDonald, Ryan, Kevin Lerman, and Fernando Pereira. 2006. Multilingual Dependency Analysis with a Two-Stage Discriminative Parser. In *CoNLL-2006: Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 216–220.
- Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *CoNLL-2007: Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL-2007*, pages 915–932.
- Pradhan, Sameer, Honglin Sun, Wayne Ward, James H. Martin, and Dan Jurafsky. 2004a. Parsing Arguments of Nominalizations in English and Chinese. In *HLT-NAACL-2004: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 141–144.
- Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James H. Martin, and Dan Jurafsky. 2004b. Shallow Semantic Parsing Using Support Vector Machines. In *HLT-NAACL-2004: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 233–240.
- Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *CoNLL-2008: Proceedings of the 12th Conference on Computational Natural Language Learning*.
- Xue, Nianwen and Martha Palmer. 2004. Calibrating Features for Semantic Role Labeling. In *EMNLP-2004: Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94.

Semantic Dependency Parsing using N-best Semantic Role Sequences and Roleset Information

Joo-Young Lee, Han-Cheol Cho, and Hae-Chang Rim

Natural Language Processing Lab.

Korea University

Seoul, South Korea

{jylee, hccho, rim}@nlp.korea.ac.kr

Abstract

In this paper, we describe a syntactic and semantic dependency parsing system submitted to the shared task of CoNLL 2008. The proposed system consists of five modules: syntactic dependency parser, predicate identifier, local semantic role labeler, global role sequence candidate generator, and role sequence selector. The syntactic dependency parser is based on Malt Parser and the sequence candidate generator is based on CKY style algorithm. The remaining three modules are implemented by using maximum entropy classifiers. The proposed system achieves 76.90 of labeled F1 for the overall task, 84.82 of labeled attachment, and 68.71 of labeled F1 on the WSJ+Brown test set.

1 Introduction

In the framework of the CoNLL08 shared task (Surdeanu et al., 2008), a system takes POS tagged sentences as input and produces sentences parsed for syntactic and semantic dependencies as output. A syntactic dependency is represented by an ID of head word and a dependency relation between the head word and its modifier in a sentence. A Semantic dependency is represented by predicate rolesets and semantic arguments for each predicate.

The task combines two sub-tasks: syntactic dependency parsing and semantic role labeling. Among the sub-tasks, we mainly focus on the semantic role labeling task. Compared to previous

CoNLL 2004 and 2005 shared tasks (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005) and other semantic role labeling research, major differences of our semantic role labeling task are 1) considering nominal predicates and 2) identifying roleset of predicates. Based on our observation that verbal predicate and nominal predicate have different characteristics, we decide to build different classification models for each predicate types. The models use same features but, their statistical parameters are different. In this paper, maximum entropy¹ is used as the classification model, but any other classification models such as Naive Bayes, SVM, etc. also can be used. To identify roleset, we investigate a roleset match scoring method which evaluate how likely a roleset is matched with the given predicate.

2 System Description

The proposed system sequentially performs syntactic dependency parsing, predicate identification, local semantic role classification, global sequence generation, and roleset information based selection.

2.1 Syntactic Dependency Parsing

In the proposed system, Malt Parser (Nivre et al., 2007) is adopted as the syntactic dependency parser. Although the training and test set of CoNLL08 use non-projective dependency grammar, we decide to use projective parsing algorithm, Nivre arc-standard, and projective/non-projective conversion functions that Malt Parser provides. The reason is that non-projective parsing shows worse performance than projective parsing with conversion in our preliminary experiment.

¹We use Zhang Le's MaxEnt toolkit, http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

We projectize the non-projective training sentences in the training set to generate projective sentences. And then, the parser is trained with the transformed sentences. Finally, the parsing result is converted into non-projective structure by using a function of Malt Parser.

2.2 Predicate Identification

Unlike previous semantic role labeling task (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005), predicates of sentences are not provided with input in the CoNLL08. It means that a system needs to identify which words in a sentence are predicates.

We limit predicate candidates to the words that exist in the frameset list of Propbank and Nombank. Propbank and Nombank provide lists of about 3,100 verbal predicates and about 4,400 nominal predicates. After dependency parsing, words which are located in the frameset list are selected as predicate candidates. The predicate identifier determines if a candidate is a predicate or not. The identifier is implemented by using two maximum entropy models, the one is for verbal predicates and the other is for nominal predicates. The following features are used for predicate identification:

Common Features For Predicate Identification

- Lemma of Previous Word
- Lemma of Current Word
- Lemma of Next Word
- POS of Previous Word
- POS of Current Word
- POS of Next Word
- Dependency Label of Previous Word
- Dependency Label of Current Word
- Dependency Label of Next Word

Additional Features for Verbal Predicate

- Lemma + POS of Current Word
- Trigram Lemma of Previous, Current, and Next Word

Additional Features for Nominal Predicate

- Lemma of Head of Current Word
 - POS of Head of Current Word
 - Dependency Label of Head of Current Word
-

Verbal predicate identifier shows 87.91 of F1 and nominal predicate identifier shows 81.58 of F1.

Through a brief error analysis, we found that main bottle neck for verbal predicate is auxiliary verb be and have.

2.3 Local Semantic Role Labeling

Prediate identification is followed by argument labeling. For the given predicate, the system first eliminates inappropriate argument candidates. The argument identification uses different strategies for verbs, nouns, and other predicates.

The argument classifier extracts features and labels semantic roles. None is used to indicate that a word is not a semantic argument. The classifier also uses different maximum entropy models for verbs, nouns, and other predicates

2.3.1 Argument Candidate Identification

As mentioned by Pradhan et al. (2004), argument identification poses a significant bottleneck to improving performance of Semantic Role Labeling system. We tried an algorithm motivated from Hacioglu (2004) which defined a *tree-structured family membership* of a predicate to identify more probable argument candidates and prune the others. However, we find that it works for verb and other predicate type, but does not work properly for noun predicate type. The main reason is due to the characteristics of arguments of noun predicates. First of all, a noun predicate can be an argument for itself, whereas a verb predicate cannot be. Secondly, dependency relation paths from a noun predicate to its arguments are usually shorter than a verb predicate. Although some dependency relation paths are long, they actually involve non-informative relations like IN, MD, or TO. Finally, major long distance relation paths could be identified by several path patterns acquired from the corpus.

Based on the above analysis, we specify a new argument identification strategy for nominal predicate type. The argument identifier regards a predicate and its nearest neighbors - its parent and children - as argument candidates. However, if the POS tag of a nearest neighbor is IN, MD, or TO, it will be ignored and the next nearest candidates will be used. Moreover, several patterns (three consecutive nouns, adjective and two consecutive nouns, two nouns combined with conjunction, and etc.) are applied to find long distance argument candidates.

2.3.2 Argument Classification

For argument classification, various features have been used. Primarily, we tested a set of features suggested by Hacıoglu (2004). The voice of the predicate, left and right words, its POS tag for a predicate, and lexical clues for adjunctive arguments also have been tested. Based on the type of predicate (i.e. verb predicate, noun predicate, and other predicate) three classification models are trained by using maximum entropy with the following same features:

Features for Argument Classification

- Dependen Relation Type
 - Family Membership
 - Position
 - Lemma of Head Word
 - POS of Head Word
 - Path
 - POS Pattern of Predicate's Children
 - Relation Pattern of Predicate's Children
 - POS Pattern of Predicate's Siblings
 - Relation Pattern of Predicate's Siblings
 - POS of candidate
 - Lemma of Left Word of Candidate
 - POS of Left Word of Candidate
 - Lemma of Right Word of Candidate
 - POS of Right Word of Candidate
-

The classifier produces a list of possible semantic roles and its probabilities for each word in the given sentence.

2.4 Global Semantic Role Sequence Generation

For local semantic role labeling, we assume that semantic roles of words are independent of each other. Toutanova et al. (2005) and Surdeanu et al. (2007) show that global constraint and optimization are important in semantic role labeling. We use CKY-based dynamic programming strategy, similar to Surdeanu et al. (2007), to verify whether role sequences satisfy global constraint and generate candidates of global semantic role sequences.

In this paper, we just use one constraint: no duplicate arguments are allowed for verbal predicates. For verbal predicates, CKY module builds a list of all kinds of combinations of semantic roles augmented with their probabilities. While building the list of semantic role sequences, it removes the

sequences that violate the global constraint. The output of CKY module is the list of semantic role sequences satisfying the global constraint.

2.5 Global Sequence Selection using Roleset Information

Finally, we need to select the most likely semantic role sequence. In addition, we need to identify a roleset for a predicate. We perform these tasks by finding a role sequence and roleset maximizing a score on the following formula:

$$\alpha \cdot c + \beta \cdot rf + \gamma \cdot mc \quad (1)$$

where, c , rf , mc are role sequence score, relative frequency of roleset, and matching score with roleset respectively. α, β, γ are tuning parameters of each factor and decided empirically by using development set. In this paper, we set α, β, γ to 0.5, 0.3, 0.2, respectively.

The role sequence score is calculated in the global semantic role sequence generation explained in Section 2.4. The relative frequency of a roleset means how many times the roleset occurred in the training set compared to the total occurrence of the predicate. It can be easily estimated by MLE.

The remaining problem is how to calculate the matching score. We use maximum entropy models as binary classifiers which output *match* and *not-match* and use probability of *match* as matching score. The features used for the roleset matching classifiers are based on following intuitions:

- If core roles (e.g., A0, A2, etc) defined in a roleset occur in a given role sequence, it seems to be the right roleset for the role sequence.
- If matched core roles are close to or have dependency relations with a predicate, it seems to be the right roleset.
- If a roleset has a particle and the predicate of a sentence also has that particle, it seems to be the right roleset. For example, the lemma of `predicate` node for the roleset `cut.05` in frameset file "cut.xml.gz" is `cut.back`, so the particle of `cut.05` is `back`. If the predicate of a sentence also has particle 'back', it seems to be the right roleset.
- If `example` node of a roleset in frameset file has a functional word for certain core role that

also exists in a given sentence, it seems to be the right roleset. For example, example node is defined as follows²:

```
<roleset id="cut.09" ...>
  <example>
    <text>
      As the building's new owner,
      Chase will have its work cut
      out for it.
    </text>
    <arg n="1">its work</arg>
    <rel>cut out</rel>
    <arg n="2" f="for">it</arg>
  </example>
</roleset>
```

Here, semantic role A2 has functional word *for*. If a given role sequence has A2 and its word is 'for', than this role sequence probably matches that roleset.

Based on these intuitions, we use following features for roleset matching:

- **Core Role Matching Count** The number of core roles exist in both roleset definition and given role sequence
- **Distance of Matched Core Role** Distance between predicate and core role which exists in both roleset and given role sequence. We use number of word and dependency path length as a distance
- **Indication for Same Particle** It becomes *yes* if given predicate and roleset have same particle. (otherwise *no*)
- **Indication for Same Functional Word** It becomes *yes* if one of core argument is same to the functional word of roleset. (otherwise *no*)

To train the roleset match classifiers, we extract semantic role sequence and its roleset from training data as a positive example. And then, we generate negative examples by changing its roleset to other roleset of that predicate. For example, the above sentence in `<text>` node³ becomes a positive example for `cut.09` and negative examples for other roleset such as `cut.01`, `cut.02`, etc.

²Some nodes are omitted to simplify the definition of example.

³Of cause, we assume that this sentence exist in training corpus. So, we will extract it from corpus, not from frameset file.

	WSJ+Brown	WSJ	Brown
LM	76.90	77.96	68.34
LA	84.82	85.69	77.83
LF	68.71	69.95	58.63

Table 1: System performance. LM, LA, LF means macro labeled F1 for the overall task, labeled attachment for syntactic dependencies, and labeled F1 for semantic dependencies, respectively

Labeled Prec.	Labeled Rec.	Labeled F1
88.68	73.89	80.28

Table 2: Performance of Local Semantic Role Labeler n WSJ test set. Gold parsing result, correct predicates, and correct rolesets are used.

3 Experimental Result

We have tested our system with the test set and obtained official results as shown in Table 1. We have also experimented on each module and obtained promising results.

We have tried to find the upper bound of the local semantic role labeling module. Table 2 shows the performance when gold syntactic parsing result, correct predicates, and correct rolesets are given. Comparing to phrase structure parser based semantic role labelings such as Pradhan et al. (2005) and Toutanova et al. (2005), our local semantic role labeler needs to enhance the performance. We will try to add some lexical features or chunk features in future works.

Next, we have analyzed the effect of roleset based selector. Table 3 shows the effect of matching score and relative frequency which are the weighted factor of selection described in section 2.5. Here, baseline means that it selects a role sequence which has the highest score in CKY module and roleset is chosen randomly. The results show that roleset matching score and relative frequency of roleset are effective to choose the correct role sequence and identify roleset.

4 Conclusion

In this paper, we have described a syntactic and semantic dependency parsing system with five different modules. Each module is developed with maximum entropy classifiers based on different predicate types. In particular, dependency relation compression method and extracted path patterns are used to improve the performance in the argu-

	Prec.	Rec.	F1
Baseline (<i>c</i>)	69.34	58.42	63.41
+ <i>mc</i>	71.40	60.20	65.32
+ <i>rf</i>	75.94	63.98	69.45
+ <i>mc, rf</i>	76.46	64.45	69.95

Table 3: Semantic scores of global sequence selection in WSJ test set. *mc, rf* means matching score and relative frequency, respectively

ment candidate identification. The roleset matching method is devised to select the most appropriate role sequence and to identify the correct roleset.

However, the current features for roleset matching seem to be not enough and other useful features are expected to be found in the future work. There is also a room for improving the method to integrate the role sequence score, matching score, and the relative frequency.

References

- Joakim Nivre, Jens Nilsson, Johan Hall, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, Erwin Marsi. 2007. MaltParser: A Language-Independent System for Data-Driven Dependency Parsing. *Natural Language Engineering*, 13(2):95–135.
- Kadri Hacioglu. 2008. Semantic role labeling using dependency trees. In *COLING '04: Proceedings of Proceedings of the 20th international conference on Computational Linguistics*. Morristown, NJ, USA.
- Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Morristown, NJ, USA.
- Mihai Surdeanu and Richard Johansson and Adam Meyers and Lluís Màrquez and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.
- Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere Comas. 2007. Combination Strategies for Semantic Role Labeling. *The Journal of Artificial Intelligence Research*, 29:105–151.
- Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Support Vector Learning for Semantic Argument Classification. *Machine Learning*. 60:11–39.
- Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow Semantic Parsing Using Support Vector Machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association of Computational Linguistics (HLT/NAACL)*. Boston, MA, USA.
- Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*.
- Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2004*.

A Cascaded Syntactic and Semantic Dependency Parsing System

Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, Sheng Li

Information Retrieval Lab

School of Computer Science and Technology

Harbin Institute of Technology, China, 150001

{car, lzh, yxhu, yqli, qinb, tliu, ls}@ir.hit.edu.cn

Abstract

We describe our CoNLL 2008 Shared Task system in this paper. The system includes two cascaded components: a syntactic and a semantic dependency parsers. A first-order projective MSTParser is used as our syntactic dependency parser. In order to overcome the shortcoming of the MSTParser, that it cannot model more global information, we add a relabeling stage after the parsing to distinguish some confusable labels, such as ADV, TMP, and LOC. Besides adding a predicate identification and a classification stages, our semantic dependency parsing simplifies the traditional four stages semantic role labeling into two: a maximum entropy based argument classification and an ILP-based post inference. Finally, we gain the overall labeled macro F1 = 82.66, which ranked the second position in the closed challenge.

1 System Architecture

Our CoNLL 2008 Shared Task (Surdeanu et al., 2008) participating system includes two cascaded components: a syntactic and a semantic dependency parsers. They are described in Section 2 and 3 respectively. Their experimental results are shown in Section 4. Section 5 gives our conclusion and future work.

2 Syntactic Dependency Parsing

MSTParser (McDonald, 2006) is selected as our basic syntactic dependency parser. It views the

syntactic dependency parsing as a problem of finding maximum spanning trees (MST) in directed graphs. MSTParser provides the state-of-the-art performance for both projective and non-projective tree banks.

2.1 Features

The score of each labeled arc is computed through the Eq. (1) in MSTParser.

$$\text{score}(h, c, l) = \mathbf{w} \cdot \mathbf{f}(h, c, l) \quad (1)$$

where node h represents the head node of the arc, while node c is the dependent node (or child node). l denotes the label of the arc.

There are three major differences between our feature set and McDonald (2006)'s:

1) We use the lemma as a generalization feature of a word, while McDonald (2006) use the word's prefix.

2) We add two new features: "bet-pos-h-same-num" and "bet-pos-c-same-num". They represent the number of nodes which locate between node h and node c and whose POS tags are the same with h and c respectively.

3) We use more back-off features than McDonald (2006) by completely enumerating all of the possible combinatorial features.

2.2 Relabeling

By observing the current results of MSTParser on the development data, we find that the performance of some labels are far below average, such as ADV, TMP, LOC. We think the main reason lies in that MSTParser only uses local features restricted to a single arc (as shown in Eq. (1)) and fails to use more global information. Consider two sentences: "I read books in the room." and "I read books in the afternoon.". It is hard to correctly label the arc

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

Deprel	Total	Mislabeled as
NMOD	8,922	NAME [0.4], DEP [0.4], LOC [0.1], AMOD [0.1]
OBJ	1,728	TMP [0.5], ADV [0.4], OPRD[0.3]
ADV	1,256	TMP [2.9], LOC [2.3], MNR [1.8], DIR [1.5]
NAME	1,138	NMOD [2.2]
VC	953	PRD [0.9]
DEP	772	NMOD [4.0]
TMP	755	ADV [9.9], LOC [6.5]
LOC	556	ADV [12.6], NMOD [7.9], TMP [5.9]
AMOD	536	ADV [2.2]
PRD	509	VC [4.7]
APPO	444	NMOD [2.5]
OPRD	373	OBJ [4.6]
DIR	119	ADV [18.5]
MNR	109	ADV [28.4]

Table 1: Error Analysis of Each Label

between “read” and “in” unless we know the object of “in”.

We count the errors of each label, and show the top ones in Table 1. “Total” refers to the total number of the corresponding label in the development data. The column of “Mislabeled as” lists the labels that an arc may be mislabeled as. The number in brackets shows the percentage of mislabeling. As shown in the table, some labels are often confusable with each other, such as ADV, LOC and TMP.

2.3 Relabeling using Maximum Entropy Classifier

We constructed two confusable label set which have a higher mutual mislabeling proportion: (NMOD, LOC, ADV, TMP, MNR, DIR) and (OBJ, OPRD). A maximum entropy classifier is used to relabel them.

Features are shown in Table 2. The first column lists local features, which contains information of the head node h and the dependent node c of an arc. “+ dir dist” means that conjoining existing features with arc direction and distance composes new features. The second column lists features using the information of node c ’s children. “word_c_c” represents form or lemma of one child of the node c . “dir_c” and “dist_c” represents the direction and distance of the arc which links node c to its child. The back-off technique is also used on these features.

Local features (+ dir dist)	Global features (+ dir_c dist_c)
word_h word_c	word_h word_c word_c_c

Table 2: Relabeling Feature Set (+ dir dist)

3 Semantic Dependency Parsing

3.1 Architecture

The whole procedure is divided into four separate stages: Predicate Identification, Predicate Classification, Semantic Role Classification, and Post Inference.

During the Predicate Identification stage we examine each word in a sentence to discover target predicates, including both noun predicates (from NomBank) and verb predicates (from PropBank). In the Predicate Classification stage, each predicate is assigned a certain sense number. For each predicate, the probabilities of a word in the sentence to be each semantic role are predicted in the Semantic Role Classification stage. Maximum entropy model is selected as our classifiers in these stages. Finally an ILP (Integer Linear Programming) based method is adopted for post inference (Punyakanok et al., 2004).

3.2 Predicate Identification

The predicate identification is treated as a binary classification problem. Each word in a sentence is predicted to be a predicate or not to be. A set of features are extracted for each word, and an optimized subset of them are adopted in our final system. The following is a full list of the features:

DEPREL (a1): Type of relation to the parent.

WORD (a21), POS (a22), LEMMA (a23), HEAD (a31), HEAD_POS (a32), HEAD_LEMMA (a33): The forms, POS tags and lemmas of a word and it’s headword (parent) .

FIRST_WORD (a41), FIRST_POS (a42), FIRST_LEMMA (a43), LAST_WORD (a51), LAST_POS (a52), LAST_LEMMA (a53): A corresponding “constituent” for a word consists of all descendants of it. The forms, POS tags and lemmas of both the first and the last words in the constituent are extracted.

POS_PAT (a6): A “POS pattern” is produced for the corresponding constituent as follows: a POS bag is produced with the POS tags of the words in the constituent except for the first and the last ones, duplicated tags removed and the original order ignored. Then we have the POS_PAT feature

by combining the POS tag of the first word, the bag and the POS tag of the last word.

CHD_POS (a71), CHD_POS_NDUP (a72), CHD_REL (a73), CHD_REL_NDUP (a74): The POS tags of the child words are joined together to form feature CHD_POS. With adjacently duplicated tags reduced to one, feature CHD_POS_NDUP is produced. Similarly we can get CHD_REL and CHD_REL_NDUP too, with the relation types substituted for the POS tags.

SIB_REL (a81), SIB_REL_NDUP (a82), SIB_POS (a83), SIB_POS_NDUP (a84): Sibling words (including the target word itself) and the corresponding dependency relations (or POS tags) are considered as well. Four features are formed similarly to those of child words.

VERB_VOICE (a9): Verbs are examined for voices: if the headword lemma is either “be” or “get”, or else the relation type is “APPO”, then the verb is considered passive, otherwise active.

Also we used some “combined” features which are combinations of single features. The final optimized feature set is (a1, a21, a22, a31, a32, a41, a42, a51, a52, a6, a72, a73, a74, a81, a82, a83, a1+a21, a21+a31, a21+a6, a21+a74, a73+a81, a81+a83).

3.3 Predicate Classification

After predicate identification is done, the resulting predicates are processed for sense classification. A classifier is trained for each predicate that has multiple senses on the training data (There are totally 962 multi-sense predicates on the training corpus, taking up 14% of all) In addition to those features described in the predicate identification section, some new ones relating to the predicate word are introduced:

BAG_OF_WORD (b11), BAG_OF_WORD_O (b12): All words in a sentence joined, namely “Bag of Words”. And an “ordered” version is introduced where each word is prefixed with a letter “L”, “R” or “T” indicating it’s to the left or right of the predicate or is the predicate itself.

BAG_OF_POS_O (b21), BAG_OF_POS_N (b22): The POS tags prefixed with “L”, “R” or “T” indicating the word position joined together, namely “Bag of POS (Ordered)”. With the prefixed letter changed to a number indicating the distance to the predicate (negative for being left to the predicate and positive for right), another feature is formed, namely “Bag of POS

(Numbered)”.

WIND5_BIGRAM (b3): 5 closest words from both left and right plus the predicate itself, in total 11 words form a “window”, within which bigrams are enumerated.

The final optimized feature set for the task of predicate classification is (a1, a21, a23, a71, a72, a73, a74, a81, a82, a83, a84, a9, b11, b12, b22, b3, a71+a9).

3.4 Semantic Role Classification

In our system, the identification and classification of semantic roles are achieved in a single stage (Liu et al., 2005) through one single classifier (actually two, one for noun predicates, and the other for verb predicates). Each word in a sentence is given probabilities to be each semantic role (including none of the these roles) for a predicate. Features introduced in addition to those of the previous subsections are the following:

POS_PATH (c11), REL_PATH (c12): The “POS Path” feature consists of POS tags of the words along the path from a word to the predicate. Other than “Up” and “Down”, the “Left” and “Right” direction of the path is added. Similarly, the “Relation Path” feature consists of the relation types along the same path.

UP_PATH (c21), UP_REL_PATH (c22): “Upstream paths” are parts of the above paths that stop at the common ancestor of a word and the predicate.

PATH_LEN (c3): Length of the paths

POSITION (c4): The relative position of a word to the predicate: Left or Right.

PRED_FAMILYSHIP (c5): “Familyship relation” between a word and the predicate, being one of “self”, “child”, “descendant”, “parent”, “ancestor”, “sibling”, and “not-relative”.

PRED_SENSE (c6): The lemma plus sense number of the predicate

As for the task of semantic role classification, the features of the predicate word in addition to those of the word under consideration can also be used; we mark features of the predicate with an extra ‘p’. For example, the head word of the current word is represented as a31, and the head word of the predicate is represented as pa31. So, with no doubt for the representation, our final optimized feature set for the task of semantic role classification is (a1, a23, a33, a43, a53, a6, c11, c12, c21, c3, c4, c6, pa23, pa71, pa73,

pa83, a1+a23+a33, a21+c5, a23+c12, a33+c12, a33+c22, a6+a33, a73+c5, c11+c12, pa71+pa73).

3.5 ILP-based Post Inference

The final semantic role labeling result is generated through an ILP (Integer Linear Programming) based post inference method. An ILP problem is formulated with respect to the probability given by the above stage. The final labeling is formed at the same time when the problem is solved.

Let W be the set of words in the sentence, and R be the set of semantic role labels. A virtual label “NULL” is also added to R , representing “none of the roles is assigned”.

For each word $w \in W$ and semantic role label $r \in R$ we create a binary variable $v_{wr} \in (0, 1)$, whose value indicates whether or not the word w is labeled as label r . p_{wr} denotes the possibility of word w to be labeled as role r . Obviously, when objective function $f = \sum_{w,r} \log(p_{wr} \cdot v_{wr})$ is maximized, we can read the optimal labeling for a predicate from the assignments to the variables v_{wr} . There are three constraints used in our system:

C1: Each relation should be and only be labeled with one label (including the virtual label “NULL”), i.e.:

$$\sum_r v_{wr} = 1$$

C2: Roles with a small probability should never be labeled (except for the virtual role “NULL”). The threshold we use in our system is 0.3, which is optimized from the development data. i.e.:

$$v_{wr} = 0, \text{ if } p_{wr} < 0.3 \text{ and } r \neq \text{“NULL”}$$

C3: Statistics shows that the most roles (except for the virtual role “NULL”) usually appear only once for a predicate, except for some rare exception. So we impose a no-duplicated-roles constraint with an exception list, which is constructed according to the times of semantic roles’ duplication for each single predicate (different senses of a predicate are considered different) and the ratio of duplication to non-duplication.

$$\sum_r v_{wr} \leq 1, \text{ if } \langle p, r \rangle \notin \{ \langle p, r \rangle \mid p \in P, r \in R; \frac{d_{pr}}{c_{pr} - d_{pr}} > 0.3 \wedge d_{pr} > 10 \} \quad (2)$$

where P is the set of predicates; c_{pr} denotes the count of words in the training corpus, which are

Predicate Type	Predicate	Label
Noun	president.01	A3
Verb	match.01	A1
Verb	tie.01	A1
Verb	link.01	A1
Verb	rate.01	A0
Verb	rate.01	A2
Verb	attach.01	A1
Verb	connect.01	A1
Verb	fit.01	A1
Noun	trader.01	SU

Table 3: No-duplicated-roles constraint exception list (obtained by Eq. (2))

labeled as $r \in R$ for predicate $p \in P$; while d_{pr} denotes something similar to c_{pr} , but what taken into account are only those words labeled with r , and there are more than one roles within the sentence for the same predicate. Table 3 lists the complete exception set, which has a size of only 10.

4 Experiments

The original MSTParser¹ is implemented in Java. We were confronted with memory shortage when trying to train a model with the entire CoNLL 2008 training data with 4GB memory. Therefore, we rewrote it with C++ which can manage the memory more exactly. Since the time was limited, we only rewrote the projective part without considering second-order parsing technique.

Our maximum entropy classifier is implemented with Maximum Entropy Modeling Toolkit². The classifier parameters: gaussian prior and iterations, are tuned with the development data for different stages respectively.

lp.solve 5.5³ is chosen as our ILP problem solver during the post inference stage.

The training time of the syntactic and the semantic parsers are 22 and 5 hours respectively, on all training data, with 2.0GHz Xeon CPU and 4G memory. While the prediction can be done within 10 and 5 minutes on the development data.

4.1 Syntactic Dependency Parsing

The experiments on development data show that relabeling process is helpful, which improves the

¹<http://sourceforge.net/projects/mstparser>

²http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

³<http://sourceforge.net/projects/lpsolve>

	Precision (%)	Recall (%)	F1
Pred Identification	91.61	91.36	91.48
Pred Classification	86.61	86.37	86.49

Table 4: The performance of predicate identification and classification

	Precision (%)	Recall (%)	F1
Simple	81.02	76.00	78.43
ILP-based	82.53	75.26	78.73

Table 5: Comparison between different post inference strategies

LAS performance from 85.41% to 85.94%. The final syntactic dependency parsing performances on the WSJ and the Brown test data are 87.51% and 80.73% respectively.

4.2 Semantic Dependency Parsing

The semantic dependency parsing component is based on the last syntactic dependency parsing component. All stages of the system are trained with the closed training corpus, while predicted against the output of the syntactic parsing.

Performance for predicate identification and classification is given in Table 4, wherein the classification is done on top of the identification.

Semantic role classification and the post inference are done on top of the result of predicate identification and classification. The final performance is presented in Table 5. A simple post inference strategy is given for comparison, where the most possible label (including the virtual label “NULL”) is select except for those duplicated non-virtual labels with lower probabilities (lower than 0.5). Our ILP-based method produces a gain of 0.30 with respect to the F1 score.

The final semantic dependency parsing performance on the development and the test (WSJ and Brown) data are shown in Table 6.

	Precision (%)	Recall (%)	F1
Development	82.53	75.26	78.73
Test (WSJ)	82.67	77.50	80.00
Test (Brown)	64.38	68.50	66.37

Table 6: Semantic dependency parsing performances

4.3 Overall Performance

The overall macro scores of our syntactic and semantic dependency parsing system are 82.38%, 83.78% and 73.57% on the development and two test (WSJ and Brown) data respectively, which is ranked the second position in the closed challenge.

5 Conclusion and Future Work

We present our CoNLL 2008 Shared Task system which is composed of two cascaded components: a syntactic and a semantic dependency parsers, which are built with some state-of-the-art methods. Through a fine tuning features and parameters, the final system achieves promising results. In order to improve the performance further, we will study how to make use of more resources and tools (open challenge) and how to do joint learning between syntactic and semantic parsing.

Acknowledgments

The authors would like to thank the reviewers for their helpful comments. This work was supported by National Natural Science Foundation of China (NSFC) via grant 60675034, 60575042, and the “863” National High-Tech Research and Development of China via grant 2006AA01Z145.

References

- Liu, Ting, Wanxiang Che, Sheng Li, Yuxuan Hu, and Huaijun Liu. 2005. Semantic role labeling system using maximum entropy classifier. In *Proceedings of CoNLL-2005*, June.
- McDonald, Ryan. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.
- Punyakanok, Vasin, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of Coling-2004*, pages 1346–1352.
- Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.

The Integration of Dependency Relation Classification and Semantic Role Labeling Using Bilayer Maximum Entropy Markov Models

Weiwei Sun and Hongzhan Li and Zhifang Sui

Institute of Computational Linguistics

Peking University

{weiwsun, lihongzhan.pku}@gmail.com, szf@pku.edu.cn

Abstract

This paper describes a system to solve the joint learning of syntactic and semantic dependencies. An directed graphical model is put forward to integrate dependency relation classification and semantic role labeling. We present a bilayer directed graph to express probabilistic relationships between syntactic and semantic relations. Maximum Entropy Markov Models are implemented to estimate conditional probability distribution and to do inference. The submitted model yields 76.28% macro-average F1 performance, for the joint task, 85.75% syntactic dependencies LAS and 66.61% semantic dependencies F1.

1 Introduction

Dependency parsing and semantic role labeling are becoming important components in many kinds of NLP applications. Given a sentence, the task of dependency parsing is to identify the syntactic head of each word in the sentence and classify the relation between the dependent and its head; the task of semantic role labeling consists of analyzing the propositions expressed by some target predicates. The integration of syntactic and semantic parsing interests many researchers and some approaches has been proposed (Yi and Palmer, 2005; Ge and Mooney, 2005). CoNLL 2008 shared task proposes the merging of both syntactic dependencies and semantic dependencies under a unique unified representation (Surdeanu et al., 2008). We explore

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

the integration problem and evaluate our approach using data provided on CoNLL 2008.

This paper explores the integration of dependency relation classification and semantic role labeling, using a directed graphical model that is also known as Bayesian Networks. The directed graph of our system can be seen as one chain of observations with two label layers: the observations are argument candidates; one layer's label set is syntactic dependency relations; the other's is semantic dependency relations. To estimate the probability distribution of each arc and do inference, we implement a Maximum Entropy Markov Model (McCallum et al., 2000). Specially, a logistic regression model is used to get the conditional probability of each arc; dynamic programming algorithm is applied to solve the "argmax" problem.

2 System Description

Our DP-SRL system consists of 5 stages:

1. dependency parsing;
2. predicate prediction;
3. syntactic dependency relation classification and semantic dependency relation identification;
4. semantic dependency relation classification;
5. semantic dependency relation inference.

2.1 Dependency Parsing

In dependency parsing stage, MSTParser¹ (McDonald et al., 2005), a dependency parser that searches for maximum spanning trees over directed graphs, is used. we use MSTParser's default

¹<http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html>

Lemma and its POS tag
Number of children
Sequential POS tags of children
Lemma and POS of Neighboring words
Lemma and POS of parent
Is the word in word list of NomBank
Is the word in word list of PropBank
Is POS of the word is VB* or NN*

Table 1: Features used to predict target predicates

parameters to train a parsing model. In the third stage of our system, dependency relations between argument candidates and target predicates are updated, if there are dependency between the candidates and the predicates.

2.2 Predicate Prediction

Different from CoNLL-2005 shared task, the target predicates are not given as input. Our system formulates the predicate predication problem as a two-class classification problem using maximum entropy classifier MaxEnt² (Berger et al., 1996). Table 1 lists features used. We use a empirical threshold to filter words: if the "being target" probability of a word is greater than 0.075, it is seen as a target predicate. This strategy achieves a 79.96% precision and a 98.62% recall.

2.3 Syntactic Dependency Relation Classification and Semantic Dependency Relation Identification

We integrate dependency parsing and semantic role labeling to some extent in this stage. Some dependency parsing systems prefer two-stage architecture: unlabeled parsing and dependency classification (Nivre et al., 2007). Previous semantic role labeling approaches also prefer two-stage architecture: argument identification and argument classification. Our system does syntactic relations classification and semantic relations identification at the same time. Specially, using a pruning algorithm, we collect a set of argument candidates; then we classify dependency relations between argument candidates and the predicates and predict whether a candidate is an argument. A directed graphical model is used to represent the relations between syntactic and semantic relations.

²http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

Lemma, POS tag voice of predicates
POS pattern of predicate's children
Is the predicate from NomBank or PropBank
Predicate class. This information is extracted from frame file of each predicate.
Position: whether the candidate is before or after the predicate
Lemma and POS tag of the candidate
Lemma and POS of Neighboring words of the candidate
Lemma and POS of sibling words of the candidate
Length of the constituent headed by the candidate
Lemma and POS of the left and right most words of the constituent of the candidate
Punctuation before and after the candidate
POS path: the chain of POS from candidate to predicate
Single Character POS path: each POS in a path is clustered to a category defined by its first character
POS Pattern (string of POS tags) of all candidates
Single Character POS Pattern of all candidates

Table 2: Features used for semantic role labeling

2.4 Semantic Dependency Relation Classification

This stage assigns the final argument labels to the argument candidates supplied from the previous stage. A multi-class classifier is trained to classify the types of the arguments supplied by the previous stage. Table 2 lists the features used. It is clear that the general type of features used here is strongly based on previous work on the SRL task (Gildea and Jurafsky, 2002; Pradhan et al., 2005; Xue and Palmer, 2004). Different from CoNLL-2005, the sense of predicates should be labeled as a part of the task. Our system assigns *01* to all predicates. This is a harsh tactic since it do not take the linguistic meaning of the argument-structure into account.

2.5 Semantic Dependency Relation Inference

The purpose of inference stage is to incorporate some prior linguistic and structural knowledge, such as "each predicate takes at most one argument of each type." We use the inference process intro-

duced by (Punyakanok et al., 2004; Koomen et al., 2005). The process is modeled as an integer Linear Programming Problem (ILP). It takes the predicted probability over each type of the arguments as inputs, and takes the optimal solution that maximizes the linear sum of the probability subject to linguistic constraints as outputs. The constraints are a subset of constraints raised by Koomen et al. (2005) and encoded as following: 1) No overlapping or embedding arguments; 2) No duplicate argument classes for A0-A5; 3) If there is an R-arg argument, then there has to be an arg argument; 4) If there is a C-arg argument, there must be an arg argument; moreover, the C-arg argument must occur after arg; 5) Given the predicate, some argument types are illegal. The list of illegal argument types is extracted from framefile.

The ILP process can improve SRL performance on constituent-based parsing (Punyakanok et al., 2004). In our experiment, it also works on dependency-based parsing.

3 Bilayer Maximum Entropy Markov Models

3.1 Sequentialization

The sequentialization of a argument-structure is similar to the pruning algorithm raised by (Xue and Palmer, 2004). Given a constituent-based parsing tree, the recursive pruning process starts from a target predicate. It first collects the siblings of the predicate; then it moves to the parent of the predicate, and collects the siblings of the parent. In addition, if a constituent is a prepositional phrase, its children are also collected.

Our system uses a similar pruning algorithm to filter out very unlikely argument candidates in a dependency-based parsing tree. Given a dependency parsing tree, the pruning process also starts from a target predicate. It first collects the dependents of the predicate; then it moves to the parent of the predicate, and collects all the dependents again. Note that, the predicate is also taken into account. If the target predicate is a verb, the process goes on recursively until it reaches the root. The process of a noun target ends when it sees a *PMOD*, *NMOD*, *SBJ* or *OBJ* dependency relation. If a preposition is returned as a candidate, its child is also collected. When the predicate is a verb, the set of constituents headed by survivors of our pruning algorithm is a superset of the set of survivors of the previous pruning algorithm on the correspond-

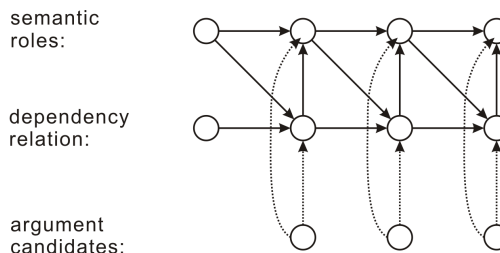


Figure 1: Directed graphical Model of The system

ing constituent-based parsing tree. This pruning algorithm will recall 99.08% arguments of verbs, and the candidates are 3.75 times of the real arguments. If the stop relation such as *PMOD* of a noun is not taken into account, the recall is 97.67% and the candidates is 6.28 times of arguments. If the harsh stop condition is implemented, the recall is just 80.29%. Since the SRL performance of nouns is very low, the harsh pruning algorithm works better than the original one.

After pruning, our system sequentializes all argument candidates of the target predicate according to their linear order in the given sentence.

3.2 Graphical Model

Figure 1 is the directed graph of our system. There is a chain of candidates $\mathbf{x} = (x_0 = BOS, x_1, \dots, x_n)$ in the graph which are observations. There are two tag layers in the graph: the up layer is information of semantic dependency relations; the down layer is information of syntactic dependency relations.

Given \mathbf{x} , denote the corresponding syntactic dependency relations $\mathbf{d} = (d_0 = BOS, d_1, \dots, d_n)$ and the corresponding semantic dependency relations $\mathbf{s} = (s_0 = BOS, s_1, \dots, s_n)$. Our system labels the syntactic and semantic relations according to the conditional probability in argmax flavor. Formally, labels the system assigned make the score $p(\mathbf{d}, \mathbf{s}|\mathbf{x})$ reaches its maximum. We decompose the probability $p(\mathbf{d}, \mathbf{s}|\mathbf{x})$ according to the directed graph modeled as following:

$$\begin{aligned}
 p(\mathbf{d}, \mathbf{s}|\mathbf{x}) &= p(s_1|s_0, d_1; \mathbf{x})p(d_1|s_0, d_0; \mathbf{x}) \cdots \\
 &\quad p(s_{i+1}|s_i, d_{i+1}; \mathbf{x})p(d_{i+1}|s_i, d_i; \mathbf{x}) \cdots \\
 &\quad p(s_n|s_{n-1}, d_n; \mathbf{x})p(d_n|s_{n-1}, d_{n-1}; \mathbf{x}) \\
 &= \prod_{i=1}^n p(s_i|s_{i-1}, d_i; \mathbf{x})p(d_i|s_{i-1}, d_{i-1}; \mathbf{x})
 \end{aligned}$$

Lemma, POS tag voice of predicates
POS pattern of predicate's children
Lemma and POS tag of the candidate
Lemma and POS of Neighboring words of the candidate
Lemma and POS of sibling words of the candidate
Length of the constituent headed by the candidate
Lemma and POS of the left and right most words of the constituent of the candidate
Conjunction of lemma of candidates and predicates; Conjunction of POS of candidates and predicates
POS Pattern of all candidates

Table 3: Features used to predict syntactic dependency parsing

3.3 Probability Estimation

The system defines the conditional probability $p(s_i|s_{i-1}, d_i; \mathbf{x})$ and $p(d_i|s_{i-1}, d_{i-1}; \mathbf{x})$ by using the maximum entropy (Berger et al., 1996) framework. Denote the tag set of syntactic dependency relations \mathcal{D} and the tag set of semantic dependency relations \mathcal{S} . Formally, given a feature map ϕ_s and a weight vector \mathbf{w}_s ,

$$p_{\mathbf{w}_s}(s_i|s_{i-1}, d_i; \mathbf{x}) = \frac{\exp\{\mathbf{w}_s \cdot \phi_s(\mathbf{x}, s_i, s_{i-1}, d_i)\}}{Z_{\mathbf{x}, s_{i-1}, d_i; \mathbf{w}_s}}$$

where,

$$Z_{\mathbf{x}, s_{i-1}, d_i; \mathbf{w}_s} = \sum_{s \in \mathcal{S}} \exp\{\mathbf{w}_s \cdot \phi_s(\mathbf{x}, s, s_{i-1}, d_i)\}$$

Similarly, given a feature map ϕ_d and a weight vector \mathbf{w}_d , ($p_{\mathbf{w}_d}(d_i)$ is short for $p_{\mathbf{w}_d}(d_i|s_{i-1}, d_{i-1}; \mathbf{x})$)

$$p_{\mathbf{w}_d}(d_i) = \frac{\exp\{\mathbf{w}_d \cdot \phi_d(\mathbf{x}, d_i, s_{i-1}, d_{i-1})\}}{Z_{\mathbf{x}, s_{i-1}, d_{i-1}; \mathbf{w}_d}}$$

where,

$$Z_{\mathbf{x}, s_{i-1}, d_{i-1}; \mathbf{w}_d} = \sum_{d \in \mathcal{D}} \exp\{\mathbf{w}_d \cdot \phi_d(\mathbf{x}, d, s_{i-1}, d_{i-1})\}$$

For different characteristic properties between syntactic parsing and semantic parsing, different feature maps are taken into account. Table 2

lists the features used to predict semantic dependency relations, whereas table 3 lists the features used to predict the syntactic dependency relations. The features used for syntactic dependency relation classification are strongly based on previous works (McDonald et al., 2006; Nakagawa, 2007).

We just integrate syntactic dependency Relation classification and semantic dependency relation here. If one combines identification and classification of semantic roles as one multi-class classification, the tag set of the second layer can be substituted by the tag set of semantic roles plus a NULL ("not an argument") label.

3.4 Inference

The "argmax problem" in structured prediction is not tractable in the general case. However, the bi-layer graphical model presented in form sections admits efficient search using dynamic programming solution. Searching for the highest probability of a graph depends on the factorization chosen. According to the form of the global score

$$p(\mathbf{d}, \mathbf{s}|\mathbf{x}) = \prod_{i=1}^n p(s_i|s_{i-1}, d_i; \mathbf{x}) p(d_i|s_{i-1}, d_{i-1}; \mathbf{x})$$

, we define forward probabilities $\alpha_t(s, d)$ to be the probability of semantic relation being s and syntactic relation being d at time t given observation sequence up to time t . The recursive dynamic programming step is

$$\alpha_{t+1}(d, s) = \arg \max_{d \in \mathcal{D}, s \in \mathcal{S}} \sum_{d' \in \mathcal{D}, s' \in \mathcal{S}} \alpha_t(d', s') \cdot p(s_i|s_{i-1}, d_i; \mathbf{x}) p(d_i|s_{i-1}, d_{i-1}; \mathbf{x})$$

Finally, to compute the globally most probable assignment $(\hat{\mathbf{d}}, \hat{\mathbf{s}}) = \arg \max_{\mathbf{d}, \mathbf{s}} p(\mathbf{d}, \mathbf{s}|\mathbf{x})$, a Viterbi recursion works well.

4 Results

We trained our system using positive examples extracted from all training data of CoNLL 2008 shared task. Table 4 shows the overall syntactic parsing results obtained on the WSJ test set (Section 23) and the Brown test set (Section ck/01-03). Table 5 shows the overall semantic parsing results obtained on the WSJ test set (Section 23) and the Brown test set (Section ck/01-03).

Test Set	UAS	LAS	Label Accuracy
WSJ	89.25%	86.37%	91.25%
Brown	86.12%	80.75%	87.14%

Table 4: Overall syntactic parsing results

	Task	Precision	Recall	$F_{\beta=1}$
WSJ	ID	73.76%	85.24%	79.08
	ID&CL	63.07%	72.88%	67.62
Brown	ID	70.77%	80.50%	75.32
	ID&CL	54.74%	62.26%	58.26

Table 5: Overall semantic parsing results

Test WSJ	Precision(%)	Recall(%)	$F_{\beta=1}$
SRL of Verbs			
All	73.53	73.28	73.41
Core-Arg	78.83	76.93	77.87
AM-*	62.51	64.83	63.65
SRL of Nouns			
All	62.06	45.49	52.50
Core-Arg	61.47	46.56	52.98
AM-*	66.19	39.93	49.81

Table 6: Semantic role labeling results on verbs and nouns. *Core-Arg* means numbered argument.

Table 6 shows the detailed semantic parsing results obtained on the WSJ test set (Section 23) of verbs and nouns respectively. The comparison suggests that SRL on NomBank is much harder than PropBank.

Acknowledgements

The work is supported by the National Natural Science Foundation of China under Grants No. 60503071, 863 the National High Technology Research and Development Program of China under Grants No.2006AA01Z144, and the Project of Toshiba (China) Co., Ltd. R&D Center.

References

Berger, Adam, Stephen Della Pietra, and Vincent Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

Ge, Ruifang and Raymond J. Mooney. 2005. A Statistical Semantic Parser that Integrates Syntax and Semantics. In *Proceedings of the Conference of Computational Natural Language Learning*.

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic

Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

Koopen, Peter, Vasina Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized Inference with Multiple Semantic Role Labeling Systems. In *Proceedings of Conference on Natural Language Learning*.

McCallum, Andrew, Dayne Freitag, and Fernando Pereira. 2000. Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Proceedings of International Conference on Machine Learning*.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

McDonald, Ryan, Kevin Lerman, and Fernando Pereira. 2006. Multilingual Dependency Analysis with a Two-Stage Discriminative Parser. In *Proceedings of Conference on Natural Language Learning*.

Nakawa, Tetsuji. 2007. Multilingual Dependency Parsing using Global Features. In *Proceedings of Conference on Natural Language Learning*.

Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. The CoNLL 2007 Shared Task on Dependency Parsing. 2007. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, 915–932.

Pradhan, Sameer, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Daniel Jurafsky. 2005. Support Vector Learning for Semantic Argument Classification. In *Proceedings of Conference on Association for Computational Linguistics*.

Punyakanok, Vasin, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic Role Labeling via Integer Linear Programming Inference. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Nivre, Joakim. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.

Xue, Nianwen and Martha Palmer. 2004. Calibrating Features for Semantic Role Labeling. In *Proceedings of Empirical Methods in Natural Language Processing*.

Yi, Szu-ting and Martha Palmer. 2005. The Integration of Syntactic Parsing and Semantic Role Labeling. In *Proceedings of the Conference of Computational Natural Language Learning*.

Mixing and Blending Syntactic and Semantic Dependencies

Yvonne Samuelsson

Dept. of Linguistics
Stockholm University
yvonne.samuelsson@ling.su.se

Oscar Täckström

Dept. of Linguistics and Philology
SICS / Uppsala University
oscar@sics.se

Sumithra Velupillai

Dept. of Computer and Systems Sciences
Stockholm University / KTH
sumithra@dsv.su.se

Johan Eklund

SSLIS
University College of Borås
johan.eklund@hb.se

Mark Fišel

Dept. of Computer Science
University of Tartu
fishel@ut.ee

Markus Saers

Dept. of Linguistics and Philology
Uppsala University
markus.saers@lingfil.uu.se

Abstract

Our system for the CoNLL 2008 shared task uses a set of individual parsers, a set of stand-alone semantic role labellers, and a joint system for parsing and semantic role labelling, all blended together. The system achieved a macro averaged labelled F_1 -score of 79.79 (WSJ 80.92, Brown 70.49) for the overall task. The labelled attachment score for syntactic dependencies was 86.63 (WSJ 87.36, Brown 80.77) and the labelled F_1 -score for semantic dependencies was 72.94 (WSJ 74.47, Brown 60.18).

1 Introduction

This paper presents a system for the CoNLL 2008 shared task on joint learning of syntactic and semantic dependencies (Surdeanu et al., 2008), combining a two-step pipelined approach with a joint approach.

In the pipelined system, eight different syntactic parses were blended, yielding the input for two variants of a semantic role labelling (SRL) system. Furthermore, one of the syntactic parses was used with an early version of the SRL system, to provide predicate predictions for a joint syntactic and semantic parser. For the final submission, all nine syntactic parses and all three semantic parses were blended.

The system is outlined in Figure 1; the dashed arrow indicates the potential for using the predi-

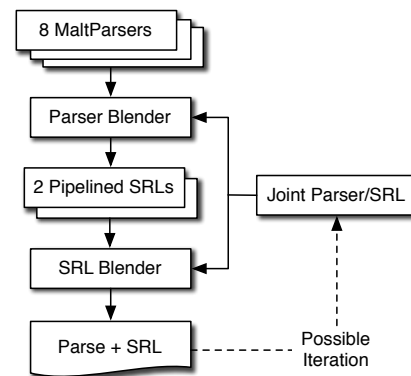


Figure 1: Overview of the submitted system.

cate prediction to improve the joint syntactic and semantic system.

2 Dependency Parsing

The initial parsing system was created using MaltParser (Nivre et al., 2007) by blending eight different parsers. To further advance the syntactic accuracy, we added the syntactic structure predicted by a joint system for syntactic and semantic dependencies (see Section 3.4) in the blending process.

2.1 Parsers

The MaltParser is a dependency parser generator, with three parsing algorithms: Nivre's arc standard, Nivre's arc eager (see Nivre (2004) for a comparison between the two Nivre algorithms), and Covington's (Covington, 2001). Both of Nivre's algorithms assume projectivity, but the MaltParser supports pseudo-projective parsing (Nilsson et al., 2007), for projectivization and de-projectivization.

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

	WSJ	Brown
Best single parse	85.22%	78.37%
LAS weights	87.00%	80.60%
Learned weights	87.36%	80.77%

Table 1: Labelled attachment score on the two test sets of the best single parse, blended with weights set to PoS labelled attachment score (LAS) and blended with learned weights.

Four parsing algorithms (the two Nivre algorithms, and Covington’s projective and non-projective version) were used, creating eight parsers by varying the parsing direction, left-to-right and right-to-left. The latter was achieved by reversing the word order in a pre-processing step and then restoring it in post-processing. For the final system, feature models and training parameters were adapted from Hall et al. (2007).

2.2 Blender

The single parses were blended following the procedure of Hall et al. (2007). The parses of each sentence were combined into a weighted directed graph. The Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967) was then used to find the maximum spanning tree (MST) of the graph, which was considered the final parse of the sentence. The weight of each graph edge was calculated as the sum of the weights of the corresponding edges in each single parse tree.

We used a simple iterative weight updating algorithm to learn the individual weights of each single parser output and part-of-speech (PoS) using the development set. To construct an initial MST, the labelled attachment score was used. Each single weight, corresponding to an edge of the hypothesis tree, was then iteratively updated by slightly increasing or decreasing the weight, depending on whether it belonged to a correct or incorrect edge as compared to the reference tree.

2.3 Results

The results are summarized in Table 1; the parse with LAS weights and the best single parse (Nivre’s arc eager algorithm with left-to-right parsing direction) are also included for comparison.

3 Semantic Role Labelling

The SRL system is a pipeline with three chained stages: *predicate identification*, *argument identification*, and *argument classification*. Predicate and

argument identification are treated as binary classification problems. In a simple post-processing predicate classification step, a predicted predicate is assigned the most frequent sense from the training data. Argument classification is treated as a multi-class learning problem, where the classes correspond to the argument types.

3.1 Learning and Parameter Optimization

For learning and prediction we used the freely available support vector machine (SVM) implementation LIBSVM (version 2.86) (Chang and Lin, 2001). The choice of cost and kernel parameter values will often significantly influence the performance of the SVM classifier. We therefore implemented a parameter optimizer based on the DIRECT optimization algorithm (Gablonsky, 2001). It iteratively divides the search space into smaller hyperrectangles, sampling the objective function in the centroid of each hyperrectangle, and selecting those hyperrectangles that are potentially optimal for further processing. The search space consisted of the SVM parameters to optimize and the objective function was the cross-validation accuracy reported by LIBSVM.

Tests performed during training for predicate identification showed that the use of runtime optimization of the SVM parameters for nonlinear kernels yielded a higher average F_1 -score effectiveness. Surprisingly, the best nonlinear kernels were always outperformed by the linear kernel with default settings, which indicates that the data is approximately linearly separable.

3.2 Filtering and Data Set Splitting

To decrease the number of instances during training, all predicate and argument candidates with PoS-tags that occur very infrequently in the training set were filtered out. Some PoS-tags were filtered out for all three stages, e.g. non-alphanumerics, HYPH, SYM, and LS. This approach was effective, e.g. removing more than half of the total number of instances for predicate prediction.

To speed up the SVM training and allow for parallelization, each data set was split into several bins. However, there is a trade-off between speed and accuracy. Performance consistently deteriorated when splitting into smaller bins. The final system contained two variants, one with more bins based on a combination of PoS-tags and lemma frequency information, and one with fewer bins

based only on PoS-tag information. The three learning tasks used different splits. In general, the argument identification step was the most difficult and therefore required a larger number of bins.

3.3 Features

We implemented a large number of features (over 50)¹ for the SRL system. Many of them can be found in the literature, starting from Gildea and Jurafsky (2002) and onward. All features, except bag-of-words, take nominal values, which are binarized for the vectors used as input to the SVM classifier. Low-frequency feature values (except for *Voice*, *Initial Letter*, *Number of Words*, *Relative Position*, and the *Distance* features), below a threshold of 20 occurrences, were given a default value.

We distinguish between single node and node pair features. The following single node features were used for all three learning tasks and for both the predicate and argument node:²

- Lemma, PoS, and Dependency relation (DepRel) for the node itself, the parent, and the left and right sibling
- Initial Letter (upper-case/lower-case), Number of Words, and Voice (based on simple heuristics, only for the predicate node during argument classification)
- PoS Sequence and PoS bag-of-words (BoW) for the node itself with children and for the parent with children
- Lemma and PoS for the first and last child of the node
- Sequence and BoW of Lemma and PoS for content words
- Sequence and BoW of PoS for the immediate children's content words
- Sequence and BoW of PoS for the parent's content words and for the parent's immediate children
- Sequence and BoW of DepRels for the node itself, for the immediate children, and for the parent's immediate children

All extractors of node pair features, where the pair consists of the predicate and the argument node, can be used both for argument identification and argument classification. We used the following node pair features:

- Relative Position (the argument is before/after the predicate), Distance in Words, Middle Distance in DepRels
- PoS Full Path, PoS Middle Path, PoS Short Path

¹Some features were discarded for the final system based on Information Gain, calculated using Weka (Witten and Frank, 2005).

²For all features using lemma or PoS the (predicted) split value is used.

The *full path* feature contains the PoS-tag of the argument node, all dependency relations between the argument node and the predicate node and finally the PoS-tag of the predicate node. The *middle path* goes to the lowest common ancestor for argument and predicate (this is also the distance calculated by *Middle Distance in DepRels*) and the *short path* only contains the dependency relation of the argument and predicate nodes.

3.4 Joint Syntactic and Semantic Parsing

When considering one predicate at a time, SRL becomes a regular labelling problem. Given a predicted predicate, joint learning of syntactic and semantic dependencies can be carried out by simultaneously assigning an argument label and a dependency relation. This is possible because we know a priori where to attach the argument, since there is only one predicate candidate³. The MaltParser system for English described in Hall et al. (2007) was used as a baseline, and then optimized for this new task, focusing on feature selection.

A large feature model was constructed, and backward selection was carried out until no further gain could be observed. The feature model of MaltParser consists of a number of feature types, each describing a starting point, a path through the structure so far, and a column of the node arrived at. The number of feature types was reduced from 37 to 35 based on the labelled F_1 -score.

As parsing is done at the same time as argument labelling, different syntactic structures risk being assigned to the same sentence, depending on which predicate is currently processed. This means that several, possibly different, parses have to be combined into one. In this experiment, the head and the dependency label were concatenated, and the most frequent one was used. In case of a tie, the first one to appear was used. The likelihood of the chosen labelling was also used as a confidence measure for the syntactic blender.

3.5 Blending and Post-Processing

Combining the output from several different systems has been shown to be beneficial (Koomen et al., 2005). For the final submission, we combined the output of two variants of the pipelined SRL system, each using different data splits, with

³The version of the joint system used in the submission was based on an early predicate prediction. More accurate predicates would give a major improvement for the results.

Test set	Pred PoS	Labelled F_1	Unlabelled F_1
WSJ	All	82.90	90.90
	NN*	81.12	86.39
	VB*	85.52	96.49
Brown	All	67.48	85.49
	NN*	58.34	75.35
	VB*	73.24	91.97

Table 2: Semantic predicate results on the test sets.

the SRL output of the joint system. A simple uniform weight majority vote heuristic was used, with no combinatorial constraints on the selected arguments. For each sentence, all predicates that were identified by a majority of the systems were selected. Then, for each selected predicate, its arguments were picked by majority vote (ignoring the systems not voting for the predicate). The best single SRL system achieved a labelled F_1 -score of 71.34 on the WSJ test set and 57.73 on the Brown test set, compared to 74.47 and 60.18 for the blended system.

As a final step, we filtered out all verbal and nominal predicates not in PropBank or NomBank, respectively, based on the predicted PoS-tag and lemma. Each lexicon was expanded with lemmas from the training set, due to predicted lemma errors in the training data. This turned out to be a successful strategy for the individual systems, but slightly detrimental for the blended system.

3.6 Results

Semantic predicate results for WSJ and Brown can be found in Table 2. Table 4 shows the results for identification and classification of arguments.

4 Analysis and Conclusions

In general, the mixed and blended system performs well on all tasks, rendering a sixth place in the CoNLL 2008 shared task. The overall scores for the submitted system can be seen in Table 3.

4.1 Parsing

For the blended parsing system, the labelled attachment score drops from 87.36 for the WSJ test set to 80.77 for the Brown test set, while the unlabelled attachment score only drops from 89.88 to 86.28. This shows that the system is robust with regards to the overall syntactic structure, even if picking the correct label is more difficult for the out-of-domain text.

The parser has difficulties finding the right head for punctuation and symbols. Apart from errors re-

	WSJ + Brown	WSJ	Brown
Syn + Sem	79.79	80.92	70.49
Syn	86.63	87.36	80.77
Sem	72.94	74.47	60.18

Table 3: Syntactic and semantic scores on the test sets for the submitted system. The scores, from top to bottom, are labelled macro F_1 , labelled attachment score and labelled F_1 .

garding punctuation, most errors occur for IN and TO. A majority of these problems are related to assigning the correct dependency. This is not surprising, since these are categories that focus on form rather than function.

There is no significant difference in score for left and right dependencies, presumably because of the bi-directional parsing. However, the system overpredicts dependencies to the root. This is mainly due to the way MaltParser handles tokens not being attached anywhere during parsing. These tokens are by default assigned to the root.

4.2 SRL

Similarly to the parsing results, the blended SRL system is less robust with respect to labelled F_1 -score, dropping from 74.47 on the WSJ test set to 60.18 on the Brown test set. The corresponding drop in unlabelled F_1 -score is from 82.90 to 75.49.

The simple method of picking the most common sense from the training data works quite well, but the difference in domain makes it more difficult to find the correct sense for the Brown corpus. In the future, a predicate classification module is needed. For the WSJ corpus, assigning the most common predicate sense works better with nominal than with verbal predicates, while verbal predicates are handled better for the Brown corpus.

In general, verbal predicate-argument structures are handled better than nominal ones, for both test sets. This is not surprising, since nominal predicate-argument structures tend to vary more in their composition.

Since we do not use global constraints for the argument labelling (looking at the whole argument structure for each predicate), the system can output the same argument label for a predicate several times. For the WSJ test set, for instance, the ratio of repeated argument labels is 5.4% in the system output, compared to 0.3% in the gold standard. However, since there are no confidence scores for predictions it is difficult to handle this in the current system.

PPOSS(pred) + ARG	WSJ F_1	Brown F_1
NN* + A0	61.42	38.99
NN* + A1	67.07	53.10
NN* + A2	57.02	26.19
NN* + A3	63.08	(16.67)
NN* + AM-ADV	4.65	(-)
NN* + AM-EXT	44.78	(40.00)
NN* + AM-LOC	49.45	(-)
NN* + AM-MNR	53.51	21.82
NN* + AM-NEG	79.37	(46.15)
NN* + AM-TMP	67.23	(25.00)
VB* + A0	81.72	73.58
VB* + A1	81.77	67.99
VB* + A2	60.91	50.67
VB* + A3	61.49	(14.28)
VB* + A4	77.84	(40.00)
VB* + AM-ADV	47.49	30.33
VB* + AM-CAU	55.12	(35.29)
VB* + AM-DIR	41.86	37.14
VB* + AM-DIS	71.91	37.04
VB* + AM-EXT	60.38	(-)
VB* + AM-LOC	55.69	37.50
VB* + AM-MNR	49.54	36.25
VB* + AM-MOD	94.85	82.42
VB* + AM-NEG	93.45	77.08
VB* + AM-PNC	50.00	(62.50)
VB* + AM-TMP	69.59	49.07
VB* + C-A1	70.76	55.32
VB* + R-A0	83.68	70.83
VB* + R-A1	68.87	51.43
VB* + R-AM-LOC	38.46	(25.00)
VB* + R-AM-TMP	56.82	(58.82)

Table 4: Semantic argument results on the two test sets, showing arguments with more than 20 instances in the gold test set (fewer instances for Brown are given in parentheses).

Acknowledgements

This project was carried out within the course Machine Learning 2, organized by GSLT (Swedish National Graduate School of Language Technology), with additional support from NGSLT (Nordic Graduate School of Language Technology). We thank our supervisors Joakim Nivre, Björn Gambäck and Pierre Nugues for advice and support. Computations were performed on the BalticGrid and UPPMAX (projects p2005008 and p2005028) resources. We thank Tore Sundqvist at UPPMAX for technical assistance.

References

- Chang, Chih-Chung and Chih-Jen Lin. 2001. *LIBSVM: A library for support vector machines*.
- Chu, Y. J. and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.
- Covington, Michael A. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the*

39th Annual Association for Computing Machinery Southeast Conference, Athens, Georgia.

- Edmonds, Jack. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71(B):233–240.
- Gablonsky, Jörg M. 2001. *Modifications of the DIRECT algorithm*. Ph.D. thesis, North Carolina State University, Raleigh, North Carolina.
- Gildea, Daniel and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.
- Hall, Johan, Jens Nilsson, Joakim Nivre, Gülşen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, Prague, Czech Republic.
- Koomen, Peter, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, Ann Arbor, Michigan.
- Nilsson, Jens, Joakim Nivre, and Johan Hall. 2007. Generalizing tree transformations for inductive dependency parsing. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.
- Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 2(13):95–135.
- Nivre, Joakim. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the ACL’04 Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, Barcelona, Spain.
- Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*, Manchester, Great Britain.
- Witten, Ian H. and Eibe Frank. 2005. *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann, Amsterdam, 2nd edition.

Dependency Tree-based SRL with Proper Pruning and Extensive Feature Engineering

Hongling Wang Honglin Wang Guodong Zhou Qiaoming Zhu

JiangSu Provincial Key Lab for Computer Information Processing Technology

School of Computer Science and Technology,

Soochow University, Suzhou, China 215006

{redleaf, 064227065055, gdzhou, qmzhu}@suda.edu.cn

Abstract

This paper proposes a dependency tree-based SRL system with proper pruning and extensive feature engineering. Official evaluation on the CoNLL 2008 shared task shows that our system achieves 76.19 in labeled macro F1 for the overall task, 84.56 in labeled attachment score for syntactic dependencies, and 67.12 in labeled F1 for semantic dependencies on combined test set, using the standalone MaltParser. Besides, this paper also presents our unofficial system by 1) applying a new effective pruning algorithm; 2) including additional features; and 3) adopting a better dependency parser, MSTParser. Unofficial evaluation on the shared task shows that our system achieves 82.53 in labeled macro F1, 86.39 in labeled attachment score, and 78.64 in labeled F1, using MSTParser on combined test set. This suggests that proper pruning and extensive feature engineering contributes much in dependency tree-based SRL.

1 Introduction

Although CoNLL 2008 shared task mainly evaluates joint learning of syntactic and semantic parsing, we focus on dependency tree-based semantic role labeling (SRL). SRL refers to label the semantic roles of predicates (either verbs or nouns) in a sentence. Most of previous SRL systems (Gildea and Jurafsky, 2002; Gildea and Palmer, 2002; Punyakanok et al., 2005; Pradhan

et al., 2004, 2005) work on constituent structure trees and has shown to achieve remarkable results. For example, Punyakanok et al. (2005) achieved the best performance in the CoNLL 2005 shared task with 79.44 in F-measure on the WSJ test set and 77.92 on the combined test set (WSJ + Brown).

With rapid development of dependency parsing in the last few years, more and more researchers turn to dependency tree-based SRL with hope to advance SRL from viewpoint of dependency parsing. Hacıoglu (2004) pioneered this work by formulating SRL as a classification problem of mapping various dependency relations into semantic roles. Compared with previous researches on constituent structure tree-based SRL which adopts constituents as labeling units, dependency tree-based SRL adopts dependency relations as labeling units. Due to the difference between constituent structure trees and dependency trees, their feature spaces are expected to be somewhat different.

In the CoNLL 2008 shared task, we extend the framework by Hacıoglu (2004) with maximum entropy as our classifier. For evaluation, we will mainly report our official SRL performance using MaltParser (Nivre and Nilsson, 2005). Besides, we will also present our unofficial system by 1) applying a new effective pruning algorithm; 2) including additional features; and 3) adopting a better dependency parser, MSTParser (McDonald, 2005).

In the remainder of this paper, we will briefly describe our system architecture, present various features used by our models and report the performance on CoNLL 2008 shared task (both official and unofficial).

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

2 System Description

In CoNLL 2008 shared task, we adopt a standard three-stage process for SRL: pruning, argument identification and argument classification. To model the difference between verb and noun predicates, we carry out separate training and testing for verb and noun predicates respectively.

In addition, we adopt OpenNLP maximum entropy package¹ in argument identification and classification.

2.1 Predicate identification

Most of Previous SRL systems only consider given predicates. However, predicates are not given in CoNLL 2008 shared task and required to be determined automatically by the system. Therefore, the first step of the shared task is to identify the verb and noun predicates in a sentence. Due to time limitation, a simple algorithm is developed to identify noun and verb predicates:

- 1) For the WSJ corpus, we simply adopt the annotations provided by PropBank and NomBank. That is, we only consider the verb and noun predicates annotated in PropBank and NomBank respectively.
- 2) For the Brown corpus, verb predicates are identified simply according to its POS tag and noun predicates are determined using a simple method that only those nouns which can also be used as verbs are identified. To achieve this goal, an English lexicon of about 56K word is applied to identify noun predicates.

Evaluation on the test set of CoNLL 2008 shared task shows that our simple predicate identification algorithm achieves the accuracies of 98.6% and 92.7 in the WSJ corpus for verb and noun predicates respectively, with overall accuracy of 95.5%, while it achieves the accuracies of 73.5% and 43.1% in the Brown corpus for verb and noun predicates respectively with overall accuracy of 61.8%. This means that the performance of predicate identification in the Brown corpus is much lower than the one in the WSJ corpus. This further suggests that much work is required to achieve reasonable predicate identification performance in future work.

2.2 Preprocessing

Using the dependency relations returned by a dependency parser (either MaltParser or

MSTParser in this paper), we can construct corresponding dependency tree for a given sentence. For example, Figure 1 shows the dependency tree of the sentence “Meanwhile, overall evidence on the economy remains fairly clouded.”. Here, W is composed of two parts: word and its POS tag with “/” as a separator while R means a dependency relation and ARG represents a semantic role.

In Hacioglu (2004), a simple pruning algorithm is applied to filter out unlikely dependency relation nodes in a dependency tree by only keeping the parent/children/grand-children of the predicate, the siblings of the predicates, and the children/grandchildren of the siblings. This paper extends the algorithm a little bit by including the nodes two more layers upward and downward with regard to the predicate’s parent, such as the predicate’s grandparent, the grandparent’s children and the grandchildren’s children. For the example as shown in Figure 1, all the nodes in the entire tree are kept. Evaluation on the training set shows that our pruning algorithm significantly reduces the training instances by 76.9%. This is at expense of wrongly pruning 1.0% semantic arguments for verb predicates. However, this figure increases to 43.5% for noun predicates due to our later observation that about half of semantic arguments of noun predicates distributes over ancestor nodes out of our consideration. This suggests that a specific pruning algorithm is necessary for noun predicates to include more ancestor nodes.

2.3 Features

Some of the features are borrowed from Hacioglu (2004) with some additional features motivated by constituent structure tree-based SRL (Pradhan et al 2005; Xue and Palmer, 2004). In the following, we explain these features and give examples with regard to the dependency tree as shown in Figure 1. We take the word *evidence* in Figure 1 as the predicate and the node “on” as the node on focus.

The following eight basic features are motivated from constituent structure tree-based SRL:

- 1) **Predicate:** predicate lemma. (evidence)
- 2) **Predicate POS:** POS of current predicate. (NN)
- 3) **Predicate Voice:** Whether the predicate (verb) is realized as an active or passive construction. If the predicate is a noun, the value is null and presented as “_”. (_)

¹https://sourceforge.net/project/showfiles.php?group_id=5961

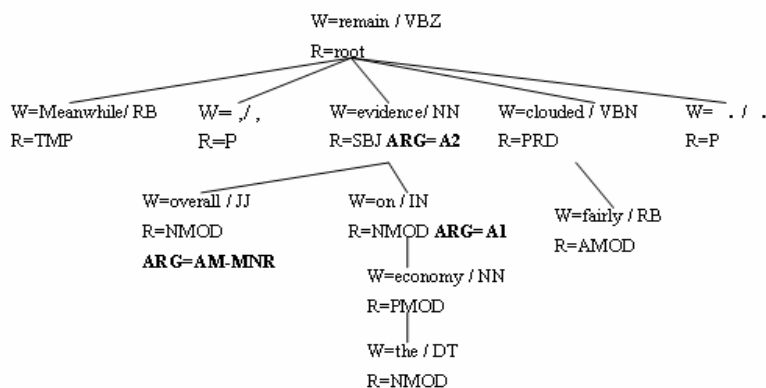


Figure 1. Example of a dependency tree augmented with semantic roles for the given predicate *evidence*.

- 4) **Relation type**: the dependency relation type of the current node. (NMOD)
 - 5) **Path**: the chain of relations from current relation node to the predicate. (NMOD->SBJ)
 - 6) **Sub-categorization**: The relation type of predicate and the left-to-right chain of the relation label sequence of the predicate's children. (SBJ->NMOD-NMOD)
 - 7) **Head word**: the head word in the relation, that is, the headword of the parent of the current node. (evidence)
 - 8) **Position**: the position of the headword of the current node with respect to the predicate position in the sentence, which can be before, after or equal. (equal)
- Besides, we also include following additional features borrowed from Hacioglu (2004):
- 1) **Family membership**: the relationship between current node and the predicate node in the family tree, such as parent, child, sibling. (child)
 - 2) **Dependent word**: the modifying word in the relation, that is, the word of current node. (on)
 - 3) **POS of headword**: the POS tag of the headword of current word. (NN)
 - 4) **POS of dependent word**: the POS tag of current word. (IN)
 - 5) **POS pattern of predicate's children**: the left-to-right chain of the POS tag sequence of the predicate's children. (JJ-IN)
 - 6) **Relation pattern of predicate's children**: the left-to-right chain of the relation label sequence of the predicate's children. (NMOD-NMOD)
 - 7) **POS pattern of predicate's siblings**: the left-to-right chain of the POS tag sequence of the predicate's siblings. (RB-.-VBN-.)

- 8) **Relation pattern of predicate's siblings**: the left-to-right chain of the relation label sequence of the predicate's siblings. (TMP-PRD-P)

3 System Performance

All the training data are included in our system, which costs 70 minutes in training and 5 seconds on testing on a PC platform with a Pentium D 3.0G CPU and 2G Memory. In particular, the argument identification stage filters out those nodes whose probabilities of not being semantic arguments are more than 0.98 for verb and noun predicates.

	Labeled Macro F1	Labeled F1	LAS
Test WSJ	78.39	70.41	85.50
Test Brown	59.89	42.67	77.06
Test WSJ+Brown	76.19	67.12	84.56

Table 1: Official performance using MaltParser (with the SRL model trained and tested on the automatic output of MaltParser)

All the performance is returned on the test set using the CoNLL 2008 evaluation script *eval08.pl* provided by the organizers. Table 1 shows the official performance using MaltParser (with the SRL model trained and tested on the automatic output of MaltParser provided by the task organizers) as the dependency parser. It shows that our system performs well on the WSJ corpus and badly on the Brown corpus largely due to bad performance on predicate identification.

4 Post-evaluation System

To gain more insights into dependency tree-based SRL, we improve the system with a new

pruning algorithm and additional features, after submitting our official results.

4.1 Effective pruning

Our new pruning algorithm is motivated by the one proposed by Xue and Palmer (2004), which only keeps those siblings to a node on the path from current predicate to the root are included, for constituent structure tree-based SRL. Our pruning algorithm further cuts off the nodes which are not related with the predicate. Besides, it filters out those nodes which are punctuations or with “symbol” dependency relations. Evaluation on the Brown corpus shows that our pruning algorithm significantly reduces the training data by 75.5% at the expense of wrongly filtering out 0.7% and 0.5% semantic arguments for verb and noun predicates respectively. This suggests that our new pruning algorithm significantly performs better than the old one in our official system, especially for the identification of noun predicates.

Furthermore, the argument identification stage filters out those nodes whose probabilities of not being semantic arguments are more than 0.90 and 0.85 for verb and noun predicates respectively, since we that our original threshold of 0.98 in the official system is too reserved.

Finally, those rarely-occurred semantic roles which occur less than 200 in the training set are filtered out and thus not considered in our system, such as A5, AA, C-A0, C-AM-ADV, R-A2 and SU.

4.2 Extensive Feature Engineering

Motivated by constituent structure tree-based SRL, two more combined features are considered in our post-evaluation system:

- 1) **Predicate + Headword:** (evidence + remain)
- 2) **Headword + Relation:** (remain + Root)

In order to better evaluate the contribution of various additional feature, we build a baseline system using hand-corrected dependency relations and the eight basic features, motivated by constituent structure tree-based SRL, as described in Section 2.3. Table 2 shows the effect of various additional features by adding one individually to the baseline system. It shows that the feature of dependent word is most useful, which improves the labeled F1 score from 81.38% to 84.84%. It also shows that the two features about predicate’s sibling deteriorate the performance. Therefore, we delete these two features from remaining experiments. Although the combined feature of “predicate+head word” is useful in constituent structure tree-based SRL, it

slightly decrease the performance in dependency tree-based SRL. For convenience, we include it in our system.

	P	R	F1
Baseline	84.31	78.64	81.38
+ Family membership	84.70	78.87	81.68
+ Dependent word	86.74	83.01	84.84
+ POS of headword	84.44	78.55	81.38
+ POS of dependent word	84.42	78.33	81.47
+ POS pattern of predicate's children	84.35	78.73	81.47
+ Relation pattern of predicate's children	84.75	78.97	81.76
+ Relation pattern of predicate's siblings	84.29	78.52	81.30
+ POS pattern of predicate's siblings	83.75	78.32	80.95
+ Predicate + Headword	83.30	78.94	81.30
+Headword + Relation	84.66	79.37	81.93

Table 2: Effects of various additional features

4.3 Best performance

Table 3 shows our system performance after applying above effective pruning strategy and additional features using the default MaltParser. Table 3 also reports our performance using the state-of-the-art MSTParser. To show the impact of predicate identification in dependency tree-based SRL, Table 4 report the performance on gold predicate identification, i.e. only using annotated predicates in the corpora.

Comparison of Table 1 and Table 3 using the MaltParser shows that our new extension with effective pruning and extensive engineering significantly improves the performance. It also shows that MSTParser-based SRL performs slightly better than MaltParser-based one, much less than the performance difference on dependency parsing between them. This suggests that such difference between these two state-of-the-art dependency parsers does not much affect corresponding SRL systems. This is also confirmed by the results in Table 4.

Comparison of Table 3 and Table 4 in labeled F1 on the Brown test data shows that the system with gold predicate identification significantly outperforms the one with automatic predicate identification using our simple algorithm by about 22 in labeled F1. This suggests that the performance of predicate identification is critical to SRL.

	MSTParser			MaltParser		
	Labeled Macro F1	Labeled F1	LAS	Labeled Macro F1	Labeled F1	LAS
Test WSJ	84.50	81.95	87.01	83.69	81.82	85.50
Test Brown	67.61	53.69	81.46	65.09	53.03	77.06
Test WSJ+Brown	82.53	78.64	86.39	81.52	78.45	84.56

Table 3: Unofficial performance using MSTParser and MaltParser with predicates automatically identified

	MSTParser			MaltParser		
	Labeled Macro F1	Labeled F1	LAS	Labeled Macro F1	Labeled F1	LAS
Test WSJ	84.75	82.45	87.01	84.04	82.52	85.50
Test Brown	78.31	75.07	81.46	75.72	74.28	77.06
Test WSJ+Brown	84.05	81.66	86.39	83.13	81.64	84.56

Table 4: Unofficial performance using MSTParser and MaltParser with gold predicate identification

5 Conclusions

This paper presents a dependency tree-based SRL system by proper pruning and extensive feature engineering. Evaluation on the CoNLL 2008 shared task shows that proper pruning and extensive feature engineering contributes much. It also shows that SRL heavily depends on the performance of predicate identification.

In future work, we will explore better ways in predicate identification. In addition, we will explore more on dependency parsing and further joint learning on syntactic and semantic parsing.

Acknowledgment

This research is supported by Project 60673041 under the National Natural Science Foundation of China and Project 2006AA01Z147 under the “863” National High-Tech Research and Development of China.

References

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28:3, pages 245-288.

Gildea, Daniel and Martha Palmer. 2002. The Necessity of Syntactic Parsing for Predicate Argument Recognition. In *Proceedings of the 40th Association for Computational Linguistics*, 2002.

Hacioglu, Kadri. 2004. Semantic Role Labeling Using Dependency Trees. In *Proceedings of the International Conference on Computational Linguistics (COLING)*. 2004.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, Jan Hajič. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *the proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 2005

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 99-106, 2005

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James H. Martin, Dan Jurafsky. 2004. Shallow Semantic Parsing Using Support Vector Machines. In *Proceedings of (HLT-NAACL-2004)*, 2004.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James H. Martin, Dan Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Association for Computational Linguistics (ACL-2005)*, 2005.

Punyakanok, Vasin, Peter Koomen, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of 9th Conference on Computational Natural Language Learning (CoNLL-2005)*. 2005

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004.

DeSRL: A Linear-Time Semantic Role Labeling System

Massimiliano Ciaramita^{†*}

massi@yahoo-inc.com

Giuseppe Attardi[‡]

attardi@di.unipi.it

Felice Dell’Orletta[‡]

dellorle@di.unipi.it mihai.surdeanu@barcelonamedia.org

Mihai Surdeanu^{†,◇}

[†]: Yahoo! Research Barcelona, Ocata 1, 08003, Barcelona, Catalunya, Spain

[‡]: Dipartimento di Informatica, Università di Pisa, L. B. Pontecorvo 3, I-56127, Pisa, Italy

[◇]: Barcelona Media Innovation Center, Ocata 1, 08003, Barcelona, Catalunya, Spain

Abstract

This paper describes the DeSRL system, a joined effort of Yahoo! Research Barcelona and Università di Pisa for the CoNLL-2008 Shared Task (Surdeanu et al., 2008). The system is characterized by an efficient pipeline of linear complexity components, each carrying out a different sub-task. Classifier errors and ambiguities are addressed with several strategies: revision models, voting, and reranking. The system participated in the closed challenge ranking third in the complete problem evaluation with the following scores: 82.06 labeled macro F1 for the overall task, 86.6 labeled attachment for syntactic dependencies, and 77.5 labeled F1 for semantic dependencies.

1 System description

DeSRL is implemented as a sequence of components of linear complexity relative to the sentence length. We decompose the problem into three sub-tasks: parsing, predicate identification and classification (PIC), and argument identification and classification (AIC). We address each of these sub-tasks with separate components without backward feedback between sub-tasks. However, the use of multiple parsers at the beginning of the process, and re-ranking at the end, contribute beneficial stochastic aspects to the system. Figure 1 summarizes the system architecture. We detail the parsing

sub-task in Section 2 and the semantic sub-tasks (PIC and AIC) in Section 3.

2 Parsing

In the parsing sub-task we use a combination strategy on top of three individual parsing models, two developed in-house –DeSR_{left-to-right} and DeSR_{right-to-left}^{revision} and a third using an off-the-shelf parser, Malt 1.0.0¹.

2.1 DeSR_{left-to-right}

This model is a version of DeSR (Attardi, 2006), a deterministic classifier-based *Shift/Reduce* parser. The parser processes input tokens advancing on the input from left to right with *Shift* actions and accumulates processed tokens on a stack with *Reduce* actions. The parser has been adapted for this year’s shared task and extended with additional classifiers, e.g., Multi Layer Perceptron and multiple SVMs.²

The parser uses the following features:

1. SPLIT_LEMMA: from tokens $-1, 0, 1, prev(0), leftChild(0), rightChild(0)$
2. PPOSS: from $-2, -1, 0, 1, 2, 3, prev(0), next(-1), leftChild(-1), leftChild(0), rightChild(-1), rightChild(0)$
3. DEPREL: from $leftChild(-1), leftChild(0), rightChild(-1)$
4. HDIST: from $-1, 0$

In the above list negative numbers refer to tokens on the stack, positive numbers to tokens in the input queue. We use the following path operators: $leftChild(x)$ refers to the leftmost child of token x , $rightChild(x)$ to the rightmost child of token x , $prev(x)$ and $next(x)$ respectively to the token preceding or following x in the sentence.

* All authors contributed equally to this work.

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

¹<http://w3.msi.vxu.se/~nivre/research/MaltParser.html>

²This parser is available for download at: <http://sourceforge.net/projects/desr/>.

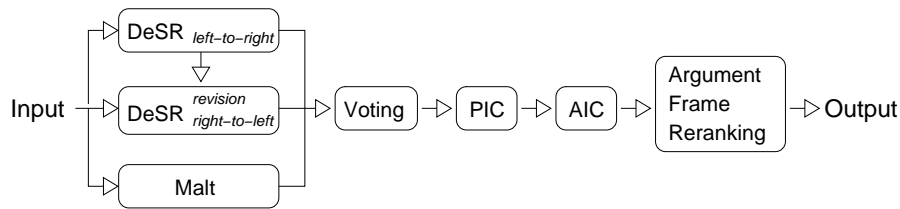


Figure 1: DeSRL system architecture.

The first three types of features are directly extracted from the attributes of tokens present in the training corpus. The fourth feature represents the distance of the token to the head of the noun phrase to which it belongs, or “O” if it does not belong to a noun phrase. This distance is computed with a simple heuristic, based on a pattern of POS tags. Attardi and Dell’Orletta (2008) have shown that this feature improves the accuracy of a shift/reduce dependency parser by providing approximate information about NP chunks in the sentence. In fact no token besides the head of a noun phrase can have a head referring to a token outside the noun phrase. Hence the parser can learn to avoid creating such links. The addition of this feature yields an increase of 0.80% in Labeled Accuracy on the development set.

2.2 Revision Parser: DeSR_{right-to-left}^{revision}

Our second individual parsing model implements an alternative to the method of revising parse trees of Attardi and Ciaramita (2007) (see also (Hall & Novak, 2005)). The original approach consisted in training a classifier to revise the errors of a baseline parser. The approach assumed that only local revisions to the parse tree would be needed, since the dependency parser mostly gets individual phrases correctly. The experiments showed that indeed most of the corrections can be expressed by a small set of (about 20) complex movement rules. Furthermore, there was evidence that one could get higher improvements from the tree revision classifier if this was trained on the output of a lower accuracy parser. The reason for this is that the number of errors is higher and this provides a larger amount of training data.

For the CoNLL 2008 shared task, we refined this idea, but instead of using an independent classifier for the revision, we use the parser itself. The second parser is trained on the original corpus extended with dependency information predicted by a lower accuracy parser. To obtain the base parser we use DeSR trained on half the training corpus using a Maximum Entropy (ME) classifier. The

ME classifier is considerably faster to train but has a lower accuracy: this model achieved an LAS of 76.49% on the development set. Using the output of the ME-based parser we extend the original corpus with four additional columns: the lemma of the predicted head (PHLEMMMA), the PPOSS of the predicted head (PHPPOSS), the dependency of the predicted head (PHDEPREL), and the indication of whether a token appears before or after its predicted head. A second parser is trained on this corpus, scanning sentences from right to left and using the following additional features:

1. PHPPOSS: from $-1, 0$
2. PHLEMMMA: from $-1, 0$
3. PHDEPREL: from $-1, 0$
4. PHHDIST: from 0

Performing parsing in reverse order helps reduce several of the errors that a deterministic parser makes when dependency links span a long distance in the input sequence. Experiments on the CoNLL 2007 corpora (Dell’Orletta, 2008) have shown that this indeed occurs, especially for distances in the range from 6 to 23. In particular, the most significant improvements are for dependencies with label COORD (+ 6%) and P (+ 8%).

The revision parser achieves an LAS of 85.81% on the development set. Note that the extra features from the forward parser are indeed useful, since a simple backward parser only achieves 82.56% LAS on the development set.

2.3 Parser Combination

The final step consists in combining the outputs of the three individual models a simple voting scheme: for each token we use majority voting to select its head and dependency label. In case of ties, we chose the dependency predicted by our overall best individual model (DeSR_{right-to-left}^{revision}).³

Note that typical approaches to parser combination combine the outputs of independent parsers, while in our case one base model (DeSR_{right-to-left}^{revision}) is trained with

³We tried several voting strategies but none performed better.

information predicted by another individual model (DeSR_{left-to-right}). To the best of our knowledge, combining individual parsing models that are inter-dependent is novel.

3 Semantic Role Labeling

We implement the Semantic Role Labeling (SRL) problem using three components: PIC, AIC, and reranking of predicted argument frames.

3.1 Predicate Identification and Classification

The PIC component carries out the identification of predicates, as well as their partial disambiguation, and it is implemented as a multiclass average Perceptron classifier (Crammer & Singer, 2003). For each token i we extract the following features ($\langle \cdot, \cdot \rangle$ stands for token combination):

1. SPLIT_LEMMA: from $\langle i-1, i \rangle, \langle i-1, i+1 \rangle, \langle i, i+1 \rangle$
2. SPLIT_FORM: from $i-2, i-1, i, i+1, i+2$
3. PPOSS: from $\langle i-2, i-1 \rangle, \langle i-1, i \rangle, \langle i-1, i+1 \rangle, \langle i, i+1 \rangle, \langle i+1, i+2 \rangle$
4. WORD_SHAPE: e.g., “Xx*” for “Brazil”, from $\langle i-2, i-1, i \rangle, \langle i-1, i \rangle, \langle i-1, i+1 \rangle, \langle i, i+1 \rangle, \langle i, i+1, i+2 \rangle$
5. Number of children of node i
6. For each children j of i : $\text{split_lemma}_j, \text{pposs}_j, \text{deprel}_{i,j}, \langle \text{split_lemma}_i, \text{split_lemma}_j \rangle, \langle \text{pposs}_i, \text{pposs}_j \rangle$
7. Difference of positions: $j-i$, for each child j of i .

The PIC component uses one single classifier mapping tokens to one of 8 classes corresponding to the rolesets suffixes 1 to 6, the 6 most frequent types, plus a class grouping all other rolesets, and a class for non predicates; i.e., $Y = \{0, 1, 2, \dots, 7\}$. Each token classified as y_7 is mapped by default to the first sense y_1 . This approach is capable of distinguishing between different predicates based on features 1 and 2, but it can also exploit information that is shared between predicates due to similar frame structures. The latter property is intuitively useful especially for low-frequency predicates.

The classifier has an accuracy in the multiclass problem, considering also the mistakes due to the non-predicted classes, of 96.2%, and an F-score of 92.7% with respect to the binary predicate identification problem. To extract features from trees (5-7) we use our parser’s output on training, development and evaluation data.

3.2 Argument Identification and Classification

Algorithm 1 describes our AIC framework. The algorithm receives as input a sentence S where predicates have been identified and classified using the

Algorithm 1: AIC

```

input : sentence  $S$ ; inference strategy  $\mathcal{I}$ ; model  $w$ 
foreach predicate  $p$  in  $S$  do
  set frame  $F_{in} = \{\}$ 
  foreach token  $i$  in  $S$  do
    if  $\text{validCandidate}(i)$  then
       $\hat{y} = \arg \max_{y \in \mathcal{Y}} \text{score}(\Phi(p, i), w, y)$ 
      if  $\hat{y} \neq \text{nil}$  then
         $\perp$  add argument  $(i, \hat{y})$  to  $F_{in}$ 
   $F_{out} = \text{inference}(F_{in}, \mathcal{I})$ 
output: set of all frames  $F_{out}$ 

```

PIC component, an inference strategy \mathcal{I} is used to guarantee that the generated best frames satisfy the domain constraints, plus an AIC classification model w . We learn w using a multiclass Perceptron, using as output label set \mathcal{Y} all argument labels that appear more than 10 times in training plus a nil label assigned to all other tokens.

During both training and evaluation we select only the candidate tokens that pass the *validCandidate* filter. This function requires that the length of the dependency path between predicate and candidate argument be less than 6, the length of the dependency path between argument and the first common ancestor be less than 3, and the length of the dependency path between the predicate and the first common ancestor be less than 5. This heuristic covers over 98% of the arguments in training.

In the worst case, Algorithm 1 has quadratic complexity in the sentence size. But, on average, the algorithm has linear time complexity because the number of predicates per sentence is small (averaging less than five for sentences of 25 words).

The function Φ generates the feature vector for a given predicate-argument tuple. Φ extracts the following features from a given tuple of a predicate p and argument a :

1. $\text{token}(a)^4$, $\text{token}(\text{modifier of } a)$ if a is the head of a prepositional phrase, and $\text{token}(p)$.
2. Patterns of PPOSS tags and DEPREL labels for: (a) the predicate children, (b) the children of the predicate ancestor across VC and IM dependencies, and (c) the siblings of the same ancestor. In all paths we mark the position of p , a and any of their ancestors.
3. The dependency path between p and a . We add three versions of this feature: just the

⁴*token* extracts the split lemma, split form, and PPOSS tag of a given token.

path, and the path prefixed with p and a 's PPOSS tags or split lemmas.

4. Length of the dependency path.
5. Distance in tokens between p and a .
6. Position of a relative to p : before or after.

We implemented two inference strategies: *greedy* and *reranking*. The greedy strategy sorts all arguments in a frame \mathbf{F}_{in} in descending order of their scores and iteratively adds each argument to the output frame \mathbf{F}_{out} only if it respects the domain constraints with the other arguments already selected. The only domain constraint we use is that core arguments cannot repeat.

3.3 Reranking of Argument Frames

The reranking inference strategy adapts the approach of Toutanova et al. (2005) to the dependency representation with notable changes in candidate selection, feature set, and learning model. For candidate selection we modify Algorithm 1: instead of storing only \hat{y} for each argument in \mathbf{F}_{in} we store the top k best labels. Then, from the arguments in \mathbf{F}_{in} , we generate the top k frames with the highest score, where the score of a frame is the product of all its argument probabilities, computed as the *softmax* function on the output of the Perceptron. In this set of candidate frames we mark the frame with the highest F_1 score as the positive example and all others as negative examples.

From each frame we extract these features:

1. Position of the frame in the set ordered by frame scores. Hence, smaller positions indicate candidate frames that the local model considered better (Marquez et al., 2007).
2. The complete sequence of arguments and predicate for this frame (Toutanova, 2005). We add four variants of this feature: just the sequence and sequence expanded with: (a) predicate voice, (b) predicate split lemma, and (c) combination of voice and split lemma.
3. The complete sequence of arguments and predicate for this frame combined with their PPOSS tags. Same as above, we add four variants of this feature.
4. Overlap with the PropBank or NomBank frame for the same predicate lemma and sense. We add the precision, recall, and F_1 score of the overlap as features (Marquez et al., 2007).
5. For each frame argument, we add the features from the local AIC model prefixed with the

	WSJ + Brown	WSJ	Brown
Labeled macro F_1	82.69	83.83	73.51
LAS	87.37	88.21	80.60
Labeled F_1	78.00	79.43	66.41

Table 1: DeSRL results in the closed challenge, for the overall task, syntactic dependencies, and semantic dependencies.

	Devel	WSJ	Brown
DeSR _{left-to-right}	85.61	86.54	79.74
DeSR _{revision}	85.81	86.19	78.91
DeSR _{right-to-left}	84.10	85.50	77.06
Voting	87.37	88.21	80.60

Table 2: LAS of individual and combined parsers.

corresponding argument label in the current frame (Toutanova, 2005).

The reranking classifier is implemented as multi-layer perceptron with one hidden layer of 5 units, trained to solve a regression problem with a least square criterion function. Previously we experimented, unsuccessfully, with a multiclass Perceptron and a ranking Perceptron. The limited number of hidden units guarantees a small computational overhead with respect to a linear model.

4 Results and Analysis

Table 1 shows the overall results of our system in the closed challenge. Note that these scores are higher than those of our submitted run mainly due to improved parsing models (discussed below) whose training ended after the deadline. The score of the submitted system is the third best for the complete task. The system throughput in our best configuration is 28 words/second, or 30 words/second without reranking. In exploratory experiments on feature selection for the re-ranking model we found that several features classes do not contribute anything and could be filtered out speeding up significantly this last SRL step. Note however that currently over 90% of the runtime is occupied by the syntactic parsers' SVM classifiers. We estimate that we can increase throughput one order of magnitude simply by switching to a faster, multiclass classifier in parsing.

4.1 Analysis of Parsing

Table 2 lists the labeled attachment scores (LAS) achieved by each parser and by their combination on the development set, the WSJ and Brown test sets. The results are improved with respect to the official run, by using a revision parser trained on the output of the lower accuracy ME parser, as

Syntax	PIC	Inference	Labeled F ₁			Unlabeled F ₁		
			Devel	WSJ	Brown	Devel	WSJ	Brown
gold	gold	greedy	88.95	90.21	84.95	93.71	94.34	93.29
predicted	gold	greedy	85.96	86.70	78.68	90.60	90.98	88.02
predicted	predicted	greedy	79.88	79.27	66.41	86.07	85.33	80.14
predicted	predicted	reranking	80.13	79.43	66.41	86.33	85.62	80.41

Table 3: Scores of the SRL component under various configurations.

	Devel	WSJ	Brown
Unlabeled F ₁	92.69	90.88	86.96
Labeled F ₁ (PIC)	87.29	84.87	71.99
Labeled F ₁ (Sense 1)	79.62	78.94	70.11

Table 4: Scores of the PIC component.

mentioned earlier. These results show that voting helps significantly (+1.56% over the best single parser) even though inter-dependent models were used. However, our simple voting scheme does not guarantee that a well-formed tree is generated, leaving room for further improvements; e.g., as in (Sagae & Lavie, 2006).

4.2 Analysis of SRL

Table 3 shows the labeled and unlabeled F₁ scores of our SRL component as we move from gold to predicted information for syntax and PIC. For the shared task setting –predicted syntax and predicted PIC– we show results for the two inference strategies implemented: greedy and reranking. The first line in the table indicates that the performance of the SRL component when using gold syntax and gold PIC is good: the labeled F₁ is 90 points for the in-domain corpus and approximately 85 points for the out-of-domain corpus. Argument classification suffers the most on out-of-domain input: there is a difference of 5 points between the labeled scores on WSJ and Brown, even though the corresponding unlabeled scores are comparable.

The second line in the table replicates the setup of the 2005 CoNLL shared task: predicted syntax but gold PIC. This yields a moderate drop of 3 labeled F₁ points on in-domain data and a larger drop of 6 points for out-of-domain data.

We see larger drops when switching to predicted PIC (line 3): 5-6 labeled F₁ points in domain and 12 points out of domain. This drop is caused by the PIC component, e.g., if a predicate is missed the whole frame is lost. Table 4 lists the scores of our PIC component, which we compare with a baseline system that assigns sense 1 to all identified predicates. The table indicates that, even though our disambiguation component improves significantly over the baseline, it performs poorly, espe-

cially on out-of-domain data. Same as SRL, the classification sub-task suffers the most out of domain (there is a difference of 15 points between unlabeled and labeled F₁ scores on Brown).

Finally, the reranking inference strategy yields only modest improvements (last line in Table 3). We attribute these results to the fact that, unlike Toutanova et al. (2005), we use only one tree to generate frame candidates, hence the variation in the candidate frames is small. Considering that the processing overhead of reranking is already large (it quadruples the runtime of our AIC component), we do not consider reranking a practical extension to a SRL system when processing speed is a dominant requirement.

References

- G. Attardi. 2006. Experiments with a Multilanguage Non-Projective Dependency Parser. In *Proc. of CoNLL-X 2006*.
- G. Attardi and M. Ciaramita. 2007. Tree Revision Learning for Dependency Parsing. In *Proc. of NAACL/HLT 2007*.
- G. Attardi, F. Dell’Orletta. 2008. Chunking and Dependency Parsing. In *Proc. of Workshop on Partial Parsing*.
- K. Crammer and Y. Singer. 2003. *Ultraconservative Online Algorithms for Multiclass Problems*. Journal of Machine Learning Research 3: pp.951-991.
- F. Dell’Orletta. 2008. Improving the Accuracy of Dependency Parsing. PhD Thesis. *Dipartimento di Informatica, Università di Pisa*, forthcoming.
- K. Hall and V. Novak. 2005. Corrective Modeling for Non-Projective Dependency Parsing. In *Proc. of IWPT*.
- L. Marquez, L. Padro, M. Surdeanu, and L. Villarejo. 2007. UPC: Experiments with Joint Learning within SemEval Task 9. In *Proc. of SemEval 2007*.
- K. Sagae and A. Lavie. 2006. Parser Combination by reparsing. In *Proc. of HLT/NAACL*.
- M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez and J. Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proc. of CoNLL-2008*.
- K. Toutanova, A. Haghghi, and C. Manning. 2005. Joint Learning Improves Semantic Role Labeling. In *Proc. of ACL*.

Probabilistic Model for Syntactic and Semantic Dependency Parsing

Enhong Chen

Department of Computer
Science, University of Sci-
ence and Technology of
China, Hefei, China
cheneh@ustc.edu.cn

Liu Shi

Department of Computer
Science, University of Sci-
ence and Technology of
China, Hefei, China
shiliu@ustc.edu

Dawei Hu

Department of Computer
Science, University of Sci-
ence and Technology of
China, Hefei, China
dwhu@mail.ustc.edu.cn

Abstract

This paper proposes a novel method to analyze syntactic dependencies and label semantic dependencies around both the verbal predicates and the nouns. In this method, a probabilistic model is designed to obtain a global optimal result. Moreover, a predicate identification model and a disambiguation model are proposed to label predicates and their senses. The experimental results obtained on the wsj and brown test sets show that our system obtains 77% of labeled macro F1 score for the whole task, 84.47% of labeled attachment score for syntactic dependency task, and 69.45% of labeled F1 score for semantic dependency task.

1 Introduction

There are two difficulties in the CoNLL 2008 shared task. One is how to label semantic role on a dependency-based representation and how to label verbal predicates and nouns. The other one is how to combine the syntactic task with the semantic task together.

On the basis of statistical analysis of labeling results, we optimize the traditional approaches of syntactic dependency parsing and semantic role labeling. Moreover, we design a predicate identification model and a disambiguation model, which will be described in section 2.3, for labeling predicates and their senses. In the disambiguation model, an exhaustion method is used to find the best sense which is corresponding to a frame of predicate. In order to obtain a global optimization result for every

sentence, a probabilistic model is designed to combine all subtasks.

The rest of this paper is organized as follows: our system is described in section 2; and section 3 reports our results on development and test sets; at last we conclude the paper in section 4.

2 A Probabilistic Model for Syntactic and Semantic Dependency Labeling

Compared with previous tasks, this shared task is more complex. It aims to merge both syntactic and semantic dependencies under a unified representation. Obviously, it can be divided into two subtasks: syntactic dependency parsing and semantic dependency labeling. For the second subtask, predicates and their senses should be labeled before semantic arguments for predicates are labeled. Since many predicates have only one sense, it is inefficient to build a multi-label classifier to classify each predicate. When a classification approach is used, it is mandatory to consider multiple senses for those predicates with only one or two senses. To prevent assigning irrelevant senses to predicates, we do not adopt classification approach. Instead, two more subtasks, i.e., predicate identification and predicate sense labeling, are introduced in this paper. The predicate sense labeling and semantic dependency labeling are performed together with a disambiguation model.

To ensure that we can get an optimal overall syntactic and semantic dependency results through integrating the above steps, a probability model is proposed. The probabilistic model is described in Equation (1), where the score P_{sent} of a sentence labeling is the combined conditional probability of its all subtasks, P_{syn} is the probability of syntactic dependency parsing, P_{pred} is the probability of predicate identification, $P_{sem}(i)$ is the probability of

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

semantic dependency labeling for the i th-predicate, and n is the number of predicates.

$$P_{\text{sent}} = P_{\text{syn}} * P_{\text{pred}} * \prod_{i=1}^n P_{\text{sem}}(i) \quad (1)$$

For each sentence, its top- N candidates using syntactic dependency parsing are obtained. Then for each candidate, predicates and semantic arguments are labeled. At last, the best one with the highest P_{sent} is chosen as final labeling result.

2.1 Syntactic Dependency Parsing

There are several approaches for syntactic dependency parsing, as demonstrated in the CoNLL 2007 shared task. A commonly used LR algorithm is applied to this task. Unlike the best-first probabilistic shift-reduce LR algorithm used by (Kenji and Jun, 2007), here a combined probability of all parsing steps is used to evaluate parsing results, and the best one is obtained as the final result. The probability of syntactic dependency parsing is defined in Equation (2).

$$P_{\text{syn}} = \prod_{i=1}^j P_{\text{act}}(i) \quad (2)$$

where $P_{\text{act}}(i)$ is the probability of every LR action act at step i , and j is the number of all steps.

As the search space of LR parser is exponential growth with the word number, the maximum size of candidate states is limited to 50.

The features that we use are similar to (Kenji and Jun, 2007). Hence we do not describe them in this paper.

2.2 Predicate Identification

In this subtask, a MaxEnt model is adopted for classification. The features we used are as follow:

- **Base info:** FORM, LEMMA, POS (GPOS if available, or is PPOS), SPLIT_FORM, SPLIT_LEMMA, PPOSS.
- **Base syntactic dependency info:**
 - Number of modifiers;
 - Number of modifiers of the previous word;
 - Number of modifiers of the next word;
 - PPOSS of left-most modifier;
 - Deprel of left-most modifier;
 - PPOSS of right-most modifier;
 - Deprel of right-most modifier.
- **Modifiers info**
 - POS list of all modifiers: if GPOS is available, POS is GPOS. Otherwise it is PPOS.
 - DEPREL list of all modifiers;
 - SPLIT_LEMMA list of all modifiers;
 - PPOSS list of all modifiers.
- **Head’s base info**

- **Head’s base syntactic dependency info**
- **Head’s modifiers info**
- **Deprel:** the syntactic dependency relation to head.
- **Word stem**
- **Stem of right-most modifier**
- **PPOSS of right-most modifier**
- **Suffix:** The suffix of the word. We use the last 3 characters as this feature.
- **Voice:** Check if the word is a verb and is passive voice.
- **Previous word info:** Check if the previous word is a predicate.
- **Pos path to ROOT:** PPOSS list from word to ROOT through the syntactic dependency path.
- **Deprel path to ROOT:** DEPREL list from word to ROOT through the syntactic dependency path.

Through statistical analysis, we find that PPOSS of nearly all predicates are in a particular category which contains NN, NNP, NNS, VB, VBD, VBG, VBN, VBP, VBZ, and JJ. Hence we ignore the words without these PPOSS to reduce the number of samples and speed up the process of training and recognition. Meanwhile, we also ignore the words having no relational frame in PropBank or NomBank.

2.3 Predicate Sense Labeling

In this subtask, we label the sense of each predicate. Different predicates are usually unrelated even if they have the same sense number, which makes us hardly use a classifier to label them. Hence, we design a disambiguation model to solve this problem.

Firstly, for each word which has been identified to be a predicate, we find out all of its probable sense forms (corresponding to the field of “PRED”). According to statistical analysis, only about 0.05% PREDs are not described in PropBank frames or NomBank frames. So it is reasonable to assume that all PREDs could be found in PropBank or NomBank. Moreover, we find that about 96% PREDs are formed as “SPLIT_LEMMA + .sense” or “LEMMA + .sense”. As a result, when a word is identified to be a predicate, we use its LEMMA and SPLIT_LEMMA to find all possible PREDs from PropBank and NomBank. Furthermore, if some special words are unsuitable for these two forms, we should convert them into their original forms first and then find their possible PREDs.

For the rest anomalous words, we build a mapping dictionary from training data.

Secondly, for each possible sense form, we label semantic argument for all words. If a word is not a semantic argument, it would be labeled as “_”. The score of the current possible sense form is calculated as the combination of all probability of each labeling. More details about semantic dependency labeling will be described in section 2.4.

Thirdly, we choose the sense form and its semantic arguments with the highest score. The above steps will be repeated until all predicates have definite senses.

2.4 Semantic Dependency Labeling

Unlike CoNLL-2005 shared task, this shared task performing Semantic Role Labeling on a dependency-based representation (DSRL). It is a novel way for SRL and the traditional SRL methods can not directly be used here. Constituent-based SRL model needs to find out all probable constituents, while DSRL only considers the semantic dependency between word and predicate. Moreover, DSRL uses syntactic dependency parsing tree instead of traditional full syntactic parsing tree. As a result, the traditional features need to be amended accordingly. The features we used are as follows:

- **Deprel**
- **Word stem**
- **POS:** if GPOS is available, POS is GPOS. Otherwise it is PPOS.
- **Stem of right-most modifier**
- **PPOSS of right-most modifier**
- **Predicate:** the FORM of predicate.
- **PPOSS of predicate**
- **Suffix of predicate**
- **Voice:** voice of predicate
- **Position:** The position of the word with respect to its predicate. It has three values, “before”, “is” and “after”, for the predicate.
- **Deprel path to predicate:** DEPREL list from word to its predicate through the syntactic dependency path.
- **Length of syntactic dependency path to predicate**
- **Sense:** the sense of predicate

Moreover, we try to find more features with frames. Since the PropBank and NomBank are available and all predicates with senses are available for this subtask. Statistical analysis shows that nearly all core semantic arguments (AA, A0, A1, A2 ...) of a predicate are described in the

frame of predicate. But it is incorrect contrarily. Based on these observations, we design features the following features for five frequently used core arguments:

- **A0 is in predicate’s frame:** Have two values: “YES” and “NO”.
- **A1 is in predicate’s frame**
- **A2 is in predicate’s frame**
- **A3 is in predicate’s frame**
- **A4 is in predicate’s frame**

Because the other core semantic arguments are rare, we do not need to design features for them. With this method, the labeling efficiency is improved while the precision almost keeps unchanged.

As the frame information has been used in features, we do not add any valency check on the labeling result.

3 Experiments and Analysis

3.1 Data and Environment

The data provided for this Closed Challenge of shared task is part of TreeBank and Brown corpus. Training set covers sections 02-21 of TreeBank. Development set covers section 24 of TreeBank. Wsj test set covers section 23 of TreeBank. Brown test set covers sections ck01, ck02, and ck03 of the Brown corpus.

The maximum entropy classifier (Berger et al, 1996) used is Le Zhang's Maximum Entropy Modeling Toolkit and the L-BFGS parameter estimation algorithm with gaussian prior smoothing (Chen and Rosenfeld, 1999). The gaussian prior is set to 2 and the iteration count is set to 500. All results we list here are post-evaluated because there are some small modifications.

The experiments are performed on a PC with AMD Athlon™ 64 x2 4400+ CPU and 2GB main memory running Microsoft Windows XP with sp2. Our system is developed using C++.

In our experimental analysis, the abbreviations used are listed as follows:

- LAS₁: Labeled attachment score
- UAS: Unlabeled attachment score
- LAS₂: Label accuracy score
- LP: Labeled precision
- LR: Labeled recall
- LF1: Labeled F1
- UP: Unlabeled precision
- UR: Unlabeled recall
- UF1: Unlabeled F1

3.2 Syntactic Dependency Parsing

We trained two LR models for syntactic dependency parsing. The first LR model uses MaxEnt classification to determine possible parser actions and their probabilities. The second LR model also uses MaxEnt classification, but parsing is performed backwards simply by reversing the sentence before parsing starts.

For a sentence, each model can label top- N candidates and calculate the probability for every result. We join these two models by finding the candidate with the highest probability from all candidates as the final result for the sentence. Table 1 shows the results of each model and joint model. We can see that the two LR models obtain similar results. The joint model can obtain better result and increase almost one percentage. The processing time of joint model is twice more than that of the two other models.

		LR Model	LR-back Model	Joint Model
dev	LAS ₁	83.05	83.38	84.43
	UAS	86.36	86.74	87.74
	LAS ₂	89.15	89.63	90.08
wsj	LAS ₁	84.84	84.06	85.48
	UAS	87.60	86.74	88.13
	LAS ₂	90.70	90.47	91.21
brown	LAS ₁	77.29	76.95	78.91
	UAS	82.75	82.61	84.38
	LAS ₂	85.00	84.82	85.76
wsj + brown	LAS ₁	84.00	83.27	84.75
	UAS	87.06	86.28	87.71
	LAS ₂	90.07	89.84	90.6
Speed (sec/sent)		0.49	0.42	0.92

Table 1: Syntactic dependency parsing results

3.3 Predicate Identification

Our predicate identification approach is described in section 2.2. We use the gold HEAD and DEPREL fields to test our approach. The results are shown in Table 2. The labeling for each sentence spends about 14ms.

	dev	wsj	brown
Precision	93.56	93.61	87.51
Recall	93.24	93.39	89.04
F1	93.40	93.50	88.27

Table 2: Predicate identification results

3.4 Semantic Dependency Labeling

Semantic dependency labeling is the last subtask. Our DSRL model uses MaxEnt classification to determine the semantic dependency between each word and its corresponding predicate.

The gold HEAD and DEPREL and PRED fields is used to test the model.

Statistical analysis shows that, for about 99% semantic argument labels, the length of syntactic dependency path from word to predicate is less than 7. So we ignore the words with the length of 7 or more.

The final results of semantic dependency labeling are shown in Table 3. The labeling for each sentence spends about 10ms.

Brown set is an out-of-domain set and wsj set is an in-domain set. Usually, the results on wsj are much better than those on brown. But here we found that the unlabeled scores are nearly the same between wsj and brown. It shows that our model performs well at unlabeled labeling on out-of-domain set, and should be improved at labeled labeling.

	dev	wsj	brown
LP	80.50	82.47	77.29
LR	70.73	73.58	67.16
LF1	75.30	77.77	71.87
UP	92.10	92.65	92.87
UR	80.92	82.65	80.69
UF1	86.15	87.36	86.35

Table 3: Semantic dependency labeling results

3.5 Overall Result

As described in section 2, we use a probabilistic model to integrate all subtasks. In the probabilistic model, syntactic dependency parsing should parse top- N candidate results. We do the rest parsing for each candidate result and get N integrated results. Then, for each integrated result, its P_{sent} is calculated and the best one is chose as the final result.

The DSRL results around verbal predicates and nouns on wsj set are shown in Table 4. It shows that verbal predicates are labeled much better than nouns.

	Unlabeled Predicate	Labeled Predicate	Labeled Semantic Arguments
NN*	87.79	79.52	58.09
VB*	96.85	80.25	73.77

Table 4: The F1 values of DSRL around verbal predicates and nouns on wsj

Table 5 shows the overall results with different N . The results are improved when N changes from 1 to 2. However, there is nearly no improvement by increasing N from 2 to 3. So N is set to be 2 in our system. Meanwhile, the effect of this approach is not obvious. We find that

there are nearly only one or two different points between the top-2 candidate dependency parsing results. This leads to that the DSRL results with these top-2 candidate results are almost the same. This is the probable reason that the approach is not much improved with the increase of N . In the future it would be necessary for us to consider the number of different points when finding the top- N dependency results.

		N=1	N=2	N=3
dev	LP	78.58	78.93	79.01
	LR	75.58	75.52	75.33
	LF1	77.05	77.19	77.13
	UP	86.56	86.95	87.07
	UR	83.04	82.94	82.75
	UF1	84.76	84.90	84.85
wsj	LP	79.41	79.76	79.96
	LR	76.67	76.59	76.49
	LF1	78.02	78.15	78.19
	UP	86.59	86.92	87.11
	UR	83.40	83.25	83.10
	UF1	84.97	85.04	85.06
brown	LP	70.52	70.95	70.79
	LR	68	67.88	67.54
	LF1	69.24	69.38	69.13
	UP	81.87	82.39	82.28
	UR	78.65	78.47	78.14
	UF1	80.23	80.39	80.16
wsj + brown	LP	78.45	78.8	78.96
	LR	75.72	75.64	75.5
	LF1	77.06	77.18	77.19
	UP	86.08	86.43	86.59
	UR	82.89	82.73	82.56
	UF1	84.45	84.54	84.53
Speed (sec/sent)		0.93	0.94	0.95

Table 5: Overall macro scores (Wsem = 0.50)

4 Conclusion

We divide this shared task into four subtasks: syntactic dependency parsing, predicate identification, predicate sense labeling and semantic dependency labeling. Then, we design a probabilistic model to combine them. The purpose of our system is to find a global optimal result for every sentence. If a syntactic dependency parsing result has the highest probability but it is unreasonable, it would be difficult to get a semantic parsing result with high probability again. Hence, a more reasonable result may be found with lower syntactic dependency parsing probability.

In our system, we have not distinguished between nouns and verbal predicates. The experimental results show that the results of verbal predicates are much better than those of nouns. In the future, it is necessary for us to deal with them separately.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No.60573077, No.60775037), Specialized Research Fund for the Doctoral Program of Higher Education (No.2007105), and Program for New Century Excellent Talents in University (No.NCET-05-0549).

References

- Berger, S. A. Della Pietra, and V. J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.
- Che Wanxiang, Ting Liu, Sheng Li, Yuxuan Hu, and Huaijun Liu. 2005. Semantic role labeling system using maximum entropy classifier. In *Proceedings of Computational Natural Language Learning (CoNLL-2005)*.
- Gildea Daniel and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.
- Duan Xiangyu, Zhao Jun and Xu Bo. 2007. Probabilistic Parsing Action Models for Multi-Lingual Dependency Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*.
- Hacioglu K. 2004. Semantic Role Labeling Using Dependency Trees. In *Proceedings of COLING-2004*.
- Johansson R. and Nugues P. 2007. Extended Constituent-to-dependency Conversion for English. In *Proceedings of NODALIDA 2007*.
- Sagae, Kenji and Tsujii, Jun'ichi. 2007. Dependency Parsing and Domain Adaptation with LR Models and Parser Ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*.
- Stanley F. Chen and Ronald Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. *Technical Report CMU-CS-99-108*.
- Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*.
- Tsai Tzong-Han, Chia-Wei Wu, Yu-Chun Lin, and Wen-Lian Hsu. 2005. Exploiting full parsing information to label semantic roles using an ensemble of me and svm via integer linear programming. In *Proceedings of Computational Natural Language Learning (CoNLL-2005)*.

Applying Sentence Simplification to the CoNLL-2008 Shared Task

David Vickrey and Daphne Koller

Stanford University

Stanford, CA 94305-9010

{dvickrey, koller}@cs.stanford.edu

Abstract

Our submission to the CoNLL-2008 shared task (Surdeanu et al., 2008) focused on applying a novel method for semantic role labeling to the shared task. Our system first simplifies each sentence to be labeled using a set of *hand-constructed* rules; the weights of the system are trained on semantic role labeling data to generate simplifications which are as useful as possible for semantic role labeling. Our system is only a semantic role labeling system, and thus did not receive a score for Syntactic Dependencies (or, by extension, a score for the complete problem). Unlike most systems in the shared task, our system took constituency parses as input. On the sub-task of semantic dependencies, our system obtained an F1 score of 76.17, the highest in the open task. In this paper we give a high-level overview of the sentence simplification system, and discuss and analyze the modifications to this system required for the CoNLL-2008 shared task.

1 Sentence Simplification

The main technical interest of our method is a sentence simplification system. This system is described in depth in (Vickrey and Koller, 2008); for lack of space, we omit many details, including a discussion of related work, from this paper.

Current semantic role labeling systems rely primarily on syntactic features in order to identify and classify roles. Features derived from a syntactic parse of the sentence have proven particularly useful (Gildea and Jurafsky, 2002). For example, the syntactic subject of “eat” is nearly always the

©2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (<http://creativecommons.org/licenses/by-nc-sa/3.0/>). Some rights reserved.

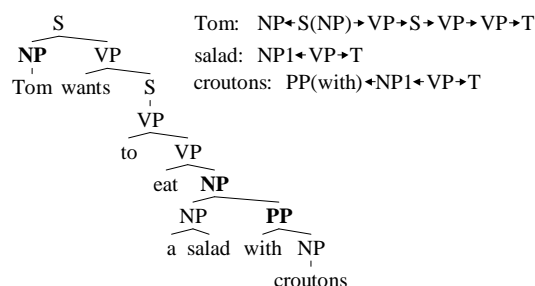


Figure 1: Constituency parse with path features for verb “eat”.

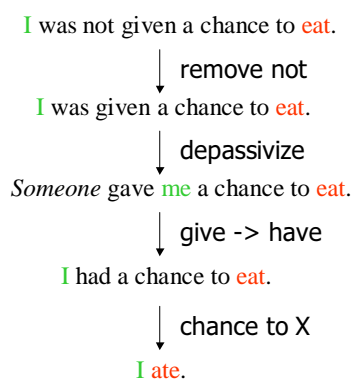


Figure 2: Example simplification

ARG0. An example sentence with extracted path features is shown in Figure 1.

A major problem with this approach is that the path from a phrase to the verb can be quite complicated. In the sentence “He expected to receive a prize for winning,” the path from “win” to its ARG0, “he”, involves the verbs “expect” and “receive” and the preposition “for.” The corresponding path through the parse tree likely occurs a small number of times (or not at all) in the training corpus. If the test set contained exactly the same sentence but with “expected” replaced by “did not expect” we would extract a different parse path feature; therefore, as far as the classifier is concerned, the syntax of the two sentences is totally unrelated.

The idea of our method is to learn a mapping from full, complicated sentences to simplified sen-

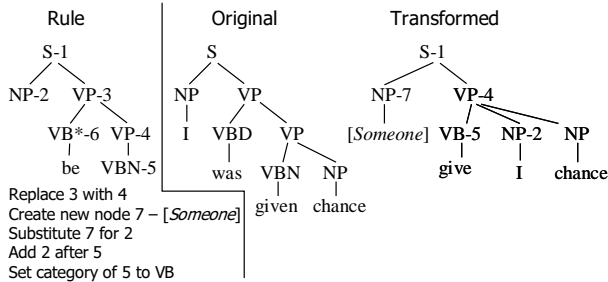


Figure 3: Rule for de-passivizing a sentence

tences. Figure 2 shows an example of a series of simplifications applied to the sentence “I was not given a chance to eat.” Our method combines hand-written syntactic simplification rules with machine learning, which determines which rules to prefer. We then use the output of the simplification system as input to an SRL system that is trained to label simplified sentences.

There are several reasons to simplify sentences before doing semantic role labeling. First, labeling simplified sentences is much easier than labeling raw sentences. Second, by mapping all sentences to a common, canonical form, we can generalize more effectively across sentences with differing syntax. Third, our model is effective at sharing information across verbs, since most of our simplification rules apply equally well regardless of the target verb. These latter two benefits are particularly important for verbs with few labeled training instances; using training examples efficiently can lead to considerable gains in performance.

Note that unlike most participants in the CoNLL-2008 Shared Task (Surdeanu et al., 2008), our model took as input constituency parses as generated by the Charniak parser (specifically, we used the parses provided with the CoNLL-2005 shared task distribution). This was the only labeled data used that was not available in the closed task.

1.1 Transformation Rules

At the center of our sentence simplification system is a hand-written set of *transformation rules*. A transformation rule takes as input a parse tree and produces as output a different, changed parse tree. Since our goal is to produce a simplified version of the sentence, the rules are designed to bring all sentences toward the same common format.

A rule (see left side of Figure 3) consists of two parts. The first is a “tree regular expression”, a tree fragment with optional constraints at each node. The rule assigns numbers to each node which are referred to in the second part of the rule. Formally, a rule node X matches a parse-tree node A if: (1)

Rule Category	#	Original	Simplified
Sentence normalization	24	Thursday, I <u>slept</u> .	I <u>slept</u> Thursday.
Sentence extraction	4	I said he <u>slept</u> .	He <u>slept</u> .
Passive	5	I was <u>hit</u> by a car.	A car <u>hit</u> me.
Misc Collapsing/Rewriting	20	John, a lawyer, ...	John is a lawyer.
Conjunctions	8	I <u>ate</u> and slept.	I <u>ate</u> .
Verb Collapsing/Rewriting	14	I must <u>eat</u> .	I <u>eat</u> .
Verb Raising/Control (basic)	17	I <u>want</u> to eat.	I <u>eat</u> .
Verb RC (ADJP/ADVP)	6	I am likely to <u>eat</u> .	I <u>eat</u> .
Verb RC (Noun)	7	I have a chance to <u>eat</u> .	I <u>eat</u> .
Modified nouns	8	The food I <u>ate</u> ...	Float(The food) I <u>ate</u> .
Floating nodes	5	Float(The food) I <u>ate</u> .	I <u>ate</u> the food.
Inverted sentences	7	Nor will I <u>eat</u> .	I will <u>eat</u> .
Questions	7	Will I <u>eat</u> ?	I will <u>eat</u> .
Possessive	7	John’s chance to <u>eat</u> ...	John has a chance to <u>eat</u> .
Verb acting as PP/NP	7	<u>Including</u> tax, the total ...	The total <u>includes</u> tax.
“Make” rewrites	8	Salt makes food <u>tasty</u> .	Food is <u>tasty</u> .

Table 1: Rule categories with sample simplifications. Target verbs are underlined.

All constraints of node X (e.g., constituent category, head word, etc.) are satisfied by node A . (2) For each child node Y of X , there is a child B of A that matches Y ; two children of X cannot be matched to the same child B . There are no other requirements. A can have other children besides those matched, and leaves of the rule pattern can match to internal nodes of the parse (corresponding to entire phrases in the original sentence). For example, the same rule is used to simplify both “I had a chance to eat,” and “I had a chance to eat a sandwich,” (into “I ate,” and “I ate a sandwich,”).

The second part of the rule is a series of simple steps that are applied to the matched nodes. For example, one type of simple step applied to the pair of nodes (X, Y) removes X from its current parent and adds it as the final child of Y . Figure 3 shows the de-passivizing rule and the result of applying it to the sentence “I was given a chance.” The transformation steps are applied sequentially from top to bottom. Any nodes not matched are unaffected by the transformation; they remain where they are relative to their parents. For example, “chance” is not matched by the rule, and thus remains as a child of the VP headed by “give.”

1.2 Rule Set

Altogether, we currently have 154 (mostly unlexicalized) rules. Table 1 shows a summary of our rule-set, grouped by type. Note that each row lists

only one possible sentence and simplification rule from that category; many of the categories handle a variety of syntax patterns. Our rule set was developed by analyzing performance and coverage on the PropBank WSJ training set; neither the development set nor (of course) the test set were used during rule creation. Please refer to (Vickrey and Koller, 2008) for more details about the rule set.

In the context of the CoNLL-2008 Shared Task, the rule set might be viewed as consisting of outside information. Since we only submitted a system to the open task, this was not an issue.

1.3 Generating Simple Sentences

We now describe how to produce all possible simplified sentences for a given input sentence. We maintain a set of derived parses S which is initialized to contain only the original, untransformed parse. One iteration of the algorithm consists of applying every possible matching transformation rule to every parse in S , and adding all resulting parses to S . With carefully designed rules, repeated iterations are guaranteed to converge; that is, we eventually arrive at a set \hat{S} such that if we apply an iteration of rule application to \hat{S} , no new parses are added. Note that we simplify the whole sentence without respect to a particular verb.

We then find all parses in \hat{S} that have “eat” as the main verb. We call such a parse a *valid simple sentence*; this is exactly the canonicalized version of the sentence which our simplification rules are designed to produce.

1.4 Labeling Simple Sentences

For a particular sentence/target verb pair s, v , the output from the previous section is a set $S^{sv} = \{t_i^{sv}\}_i$ of valid simple sentences. From the training set, we now extract a set of *role patterns* $G^v = \{g_j^v\}_j$ for each verb v . For example, a common role pattern for “give” is that of “I gave him a sandwich”. We represent this pattern as $g_1^{give} = \{ARG0 = Subject NP, ARG1 = Postverb NP2, ARG2 = Postverb NP1\}$.

For each simple sentence $t_i^{sv} \in S^{sv}$, we apply all extracted role patterns g_j^v to t_i^{sv} , obtaining a set of possible role labelings. We call a simple sentence/role labeling pair a *simple labeling* and denote the set of candidate simple labelings $C^{sv} = \{c_k^{sv}\}_k$.

1.5 Probabilistic Model

Given a (possibly large) set of candidate simple labelings C^{sv} , we need to select a correct one. We

```

Rule = Depassivize
Pattern = { ARG0 = Subj NP, ARG1 = PV NP2, ARG2 = PV NP1 }
Role = ARG0, Head Word = John
Role = ARG1, Head Word = sandwich
Role = ARG2, Head Word = I
Role = ARG0, Category = NP
Role = ARG1, Category = NP
Role = ARG2, Category = NP
Role = ARG0, Position = Subject NP
Role = ARG1, Position = Postverb NP2
Role = ARG2, Position = Postverb NP1

```

Figure 4: Features for “John gave me a sandwich.”

assign a score to each candidate based on its features: which rules were used to obtain the simple sentence, which role pattern was used, and features about the assignment of constituents to roles. The set of extracted features for the sentence “I was given a sandwich by John” with simplification “John gave me a sandwich” is shown in Figure 4.

We now define a log-linear model which assigns a probability to each candidate simple labeling based on its score. Specifically, the probability of a simple labeling c_k^{sv} with respect to a weight vector \mathbf{w} is $P(c_k^{sv}) = \frac{e^{\mathbf{w}^T \mathbf{f}_k^{sv}}}{\sum_{k'} e^{\mathbf{w}^T \mathbf{f}_{k'}^{sv}}}$.

Unfortunately, we do not have labeled examples of correct simplifications. To get around this, we treat the correct simplification as a hidden variable. Thus, we say that the probability of a particular semantic role labeling is $\sum_{c_k^{sv} \in K^{sv}} P(c_k^{sv})$. This leads to our final objective,

$$\sum_{s,v} \left(\log \frac{\sum_{c_k^{sv} \in K^{sv}} e^{\mathbf{w}^T \mathbf{f}_k^{sv}}}{\sum_{c_{k'}^{sv} \in C^{sv}} e^{\mathbf{w}^T \mathbf{f}_{k'}^{sv}}} \right) - \frac{\mathbf{w}^T \mathbf{w}}{2\sigma^2}.$$

We train our model by optimizing the objective using standard methods, specifically BFGS. Due to the summation over the hidden variable representing the choice of simplification (not observed in the training data), our objective is not convex. Thus, we are not guaranteed to find a global optimum; in practice we have gotten good results using the initialization of setting all weights to 0.

2 Baseline Model

In addition to our simplification system, we also built a high-performing logistic regression model for semantic role labeling, which we refer to as **Baseline**. This model uses a slightly modified version of the features used in (Pradhan et al., 2005). This model was also trained on the PropBank training set, using Charniak constituency parses.

Both our simplification model and **Baseline** produce labeled constituencies. Since we were required to produce semantic dependency relations, we simply labeled the head word of each constituent with the role chosen by the model.

3 Labeling Nouns

The 2008 shared task requires systems to label the arguments of both nouns and verbs. However, our sentence simplification system was built to handle only verbs. While in principle the model can naturally be extended to label nouns by the addition of further syntactic simplification rules, we were not able to complete this extension in time for the contest deadline. Instead, we trained our **Baseline** model to label the arguments of nouns as well as verbs. The features of this model are the same as those used to label verbs, and were not extended to handle special features of nouns.

4 Identifying Predicates

Another important subtask was to identify the predicates to be labeled. In the labeled training corpus, nouns with no labeled arguments are generally skipped (i.e., not labeled as predicates at all). Thus, we made a strong simplifying assumption: if a predicate (either noun or verb) is labeled by our system as having no arguments, we should not label it as being a predicate. On the development set, out of a total of 6390 labeled predicates, only 23 had no labeled arguments; thus, this assumption appears to be quite reasonable.

Our system architecture was as follows. First, we modified the training (and test) set by marking as a potential predicate every word that was either: a) a verb that wasn't "do", "be", or "have" or b) a noun found in the *nombank* index. Then, we trained our system on *all* potential predicates (not just predicates that were actually labeled). Finally, after applying our classifier to the test data, we removed any predicate with no labeled arguments.

5 Sense-Tagging Predicates

We tried three simple heuristics for sense-labeling the predicates. All of them were applied at the *end* of our pipeline, and thus did not interact with the argument labeling decisions.

The simplest heuristic labeled every predicate as sense 1. A slightly more intelligent heuristic labeled every predicate with its most common sense in the training set. Finally, we extended this heuristic to label each verb with its most common sense

for the subcategorization (i.e., set of roles) actually produced by the labeling system. Thus, if one sense was intransitive while the other was transitive, we would be able to distinguish between them (assuming that our labeling system produced the correct set of arguments). For this third heuristic, we ignored all but the core arguments (ARG0 - ARG5). The final heuristic was the most effective, as discussed in the results section.

6 Results

The first stage of **Baseline**, which tries to filter out constituents which are obviously not arguments, took about three hours and approximately 4Gb of memory to train¹. The second stage, which performs the final classification of arguments, took about four hours and 3Gb of memory to train.

The sentence simplification system, which we will refer to as **Simplification**, works in two steps. First, it generates the set of all possible simplifications for each sentence. This step took a relatively small amount of memory, under 1Gb, but took around 24 hours to complete. The set of simplifications is stored in a compact form; the total storage required for all simplifications of all sentences was roughly 4 times the (compressed) size of the Charniak input parses. The second step, which trains the model using the possible simplifications, took around 12 hours and 3Gb of memory.

We only submitted results for the semantic dependencies portion of the competition. The system we used was the **Combined** system described in (Vickrey and Koller, 2008), which combines the simplification procedure with the **Baseline** model. The **Combined** model was augmented with the modifications described above. Our system achieved an official F1 score on the SRL subtask of 76.17, the highest in the open task. Our results are not strictly comparable to those in the closed task, due to the use of the Charniak parser trained on Penn Treebank constituency parses. However, a comparison still provides insight into the relative strength of our system; our score would place us tied for fourth in the closed challenge for semantic dependencies.

We will now discuss the relative contributions of various components of our system. All results in this section are for TestWSJ + TestBrown.

Our **Combined** model provides the same benefit over **Baseline** as described in (Vickrey and

¹All runs were done on a dual core 2.66Mhz Xeon machine with 4Gb of RAM

Koller, 2008) for labeling the arguments of verbs². When applied to just verb predicates, the **Combined** model provides a statistically significant improvement of 1.2 points of F1 score over **Baseline**. However, since the CoNLL-2008 shared task adds both labeling of noun dependencies and predicate identification and sense tagging, the gain due to better labeling arguments of verbs is reduced. The **Baseline** model achieves an F1 score of 75.31 on the semantic dependencies task, .86 F1 points lower than the **Combined** system.

Note that while most of this gain is directly due to better verb argument labeling, better verb argument labeling also indirectly slightly improves predicate identification and sense-tagging since we use the predicted arguments for both of these subtasks. We do in fact see a small increase for labeling and sense-tagging predicates, from 80.72 F1 for the **Baseline** to 80.81 F1 for **Combined**.

As mentioned, we use **Baseline** to label the arguments of nouns. Noun argument labeling appears to be more difficult than verb argument labeling, or at least requires some modification of the features. **Baseline** obtains an F1 score of 75.64 for verbs, but only 68.19 F1 for nouns.

On the subtask of predicate identification, **Combined** achieved an F1 of 90.65. It performed better on verbs than nouns. For predicates with part of speech VB*³ it scored 95.43 F1; for predicates with part of speech NN*, it scored 85.97 F1. Verbs without arguments are often labeled in the gold data, so the verb score could perhaps be improved by retaining verb predicates without arguments.

As described above, we tried three heuristics for sense-labeling predicates. Our final system used the third heuristic, which chose the most common sense for the set of labeled arguments produced by the system. **Combined** obtained an F1 score of 80.81 on the combined predicate identification/classification task, with a score of 82.58 for verbs and 79.28 for nouns. The decrease in performance by adding classification is *much* larger for verbs than nouns; verb sense classification is apparently significantly more difficult than noun sense classification (at least for verbal nouns).

Table 2 compares the results of the **Combined** system using each of the three heuristics. Going

²Note that the scoring metrics are different between the CoNLL-2005 and CoNLL-2008 shared tasks. The CoNLL-2005 required the constituent boundaries to be labeled correctly, while the CoNLL-2008 only requires identifying the head word of each argument.

³This category includes some nouns, e.g. gerunds.

Heuristic	Overall Score	Predicate ID/Class		
		All	Verbs	Nouns
Always 1	75.69	79.29	81.26	77.58
Most common	76.02	80.33	81.73	79.21
Best for subcat	76.17	80.81	82.58	79.28

Table 2: Relative performance of sense-labeling heuristics

from the simplest heuristic to the third heuristic gained 1.52 points of F1 score on the subtask of predicate identification/classification, and an improvement of .48 F1 score for the overall semantic dependency score. Another interesting thing to note is that all of improvement for noun predicates came from choosing the most common sense instead of always choosing sense 1. On the other hand, using subcategorization information is quite important for sense-tagging verbs.

7 Discussion and Future Work

The CoNLL-08 task introduces two new subtasks for labeling semantic dependencies: predicate identification and predicate classification. Our experimental results show that both are non-trivial and suggest that there is room for additional improvement on these subtasks.

We are particularly interested in two extensions to our simplification model related to the 2008 shared task. The first is extending our simplification model to handle the arguments of nouns. As discussed above, there is a large amount of room for improvement for argument labeling of nouns. The second is incorporating uncertainty from the parser into our model. Specifically, we would like to extract a complete parse forest from the Charniak parser and use it as input to our model. This would allow our simplification model to jointly reason about the correct parse, possible simplifications of those parses, and semantic role labelings of the resulting simplified sentences.

References

- Gildea, D. and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*.
- Pradhan, S., K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning*, 60(1-3):11–39.
- Surdeanu, M., R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL*.
- Vickrey, D. and D. Koller. 2008. Sentence simplification for semantic role labeling. *Proceedings of ACL*.

Author Index

- Alishahi, Afra, 57
Asahara, Masayuki, 228
Attardi, Giuseppe, 258
- Bangalore, Srinivas, 151
Bianco, Maryse, 73
Blanchard, Daniel, 65
- Cappe, Olivier, 41
Carreras, Xavier, 9
Che, Wanxiang, 238
Chen, Enhong, 263
Cho, Han-Cheol, 233
Ciaramita, Massimiliano, 258
Cleuziou, Guillaume, 97
Collins, Michael, 9
Connor, Michael, 81
- Daelemans, Walter, 208
Dell'Orletta, Felice, 258
Dessus, Philippe, 73
Di Fabbrizio, Giuseppe, 151
Dias, Gaël, 97
Dyer, Charles R., 119
- Eklund, Johan, 248
Eldawy, Mohamed, 119
- Fan, Shixi, 218
Fazly, Afsaneh, 57, 89
Fishel, Mark, 248
Fisher, Cynthia, 81
- Gertner, Yael, 81
Goldberg, Andrew B., 119
Gomez, Fernando, 105
- Han, Lei, 113
Heinz, Jeffrey, 65
Henderson, James, 178
Heng, Lijie, 119
Ho, Tu Bao, 143
Hu, Dawei, 263
Hu, Yuxuan, 238
- Idiart, Marco, 49
- Iwakura, Tomoya, 17
Iwatate, Masakazu, 228
- Ji, Donghong, 113
Johansson, Richard, 159, 183
- Kate, Rohit, 33
Kit, Chunyu, 203
Koller, Daphne, 268
Koo, Terry, 9
- Lee, John, 127
Lee, Joo-Young, 233
Lemaire, Benoît, 73
Li, Hongzhan, 243
Li, Lu, 218
Li, Sheng, 238
Li, Yongqiang, 238
Li, Zhenghua, 238
Liu, Ting, 238
Lluís, Xavier, 188
- Marchal, Harmony, 73
Màrquez, Lluís, 188
Màrquez, Lluís, 159
Matsumoto, Yuji, 228
Merlo, Paola, 1, 178
Meyers, Adam, 159
Meza-Ruiz, Ivan, 193
Misra, Hemant, 41
Morante, Roser, 208
Moura, Leonardo, 49
Mukelov, Raycho, 97
Musillo, Gabriele, 1, 178
- Neumann, Günter, 213
Nguyen, Minh Le, 143
Nguyen, Thai Phuong, 143
Nguyen, Thuy Linh, 135
Nguyen, Vinh Van, 143
Nivre, Joakim, 159
Nugues, Pierre, 183
- Okamoto, Seishi, 17
Ovrelid, Lilja, 25

Parisien, Christopher, 89

Qin, Bing, 238

Ramisch, Carlos, 49

Ren, Han, 113

Riedel, Sebastian, 193

Rim, Hae-Chang, 233

Roth, Dan, 81

Saers, Markus, 248

Samuelsson, Yvonne, 248

Schwartz, Hansen A., 105

Shi, Liu, 263

Shimazu, Akira, 143

Stent, Amanda, 151

Stevenson, Suzanne, 57, 89

Sui, Zhifang, 243

Sun, Weiwei, 243

Surdeanu, Mihai, 159, 258

Täckström, Oscar, 248

Titov, Ivan, 178

Ural, Ahmet Engin, 223

Uszkoreit, Hans, 198

Van Asch, Vincent, 208

Velupillai, Sumithra, 248

Vickrey, David, 268

Villavicencio, Aline, 49

Vogel, Stephan, 135

Volokh, Alexander, 213

Wan, Jing, 113

Wang, Honglin, 253

Wang, Hongling, 253

Wang, Rui, 198

Wang, Xiaolong, 218

Wang, Xuan, 218

Watanabe, Yotaro, 228

Yatbaz, Mehmet Ali, 223

Yuret, Deniz, 223

Yvon, Francois, 41

Zhang, Yi, 198

Zhao, Hai, 203

Zhou, Guodong, 253

Zhu, Qiaoming, 253

Zhu, Xiaojin, 119