CoNLL-X

Proceedings of the

Tenth Conference on Computational Natural Language Learning

8-9 June 2006
New York City, USA

Production and Manufacturing by
Omnipress Inc.
2600 Anderson Street
Madison, WI 53704

Order copies of this and other ACL proceedings from:

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

Foreword

CoNLL has turned ten! With a mix of pride and amazement over how time flies, we now celebrate the tenth time that ACL's special interest group on natural language learning, SIGNLL, holds its yearly conference.

Having a yearly meeting was the major pillar of the design plan for SIGNLL, drawn up by a circle of enthusiastic like-minded people around 1995, headed by first president David Powers and first secretary Walter Daelemans. The first CoNLL was organized as a satellite event of ACL-97 in Madrid, in the capable hands of Mark Ellison. Since then, no single year has gone by without a CoNLL. The boards of SIGNLL (with consecutive presidents Michael Brent, Walter Daelemans, and Dan Roth) have made sure that CoNLL toured the world; twice it was held in the Asian-Pacific part of the world, four times in Europe, and four times in the North-American continent.

Over time, the field of computational linguistics got to know CoNLL for its particular take on empirical methods for NLP and the ties these methods have with areas outside the focus of the typical ACL conference. The image of CoNLL was furthermore boosted by the splendid concept of the shared task, the organized competition that tackles timely tasks in NLP and has produced both powerful and sobering scientific insights. The CoNLL shared tasks have produced benchmark data sets and results on which a significant body of work in computational linguistics is based nowadays. The first shared task was organized in 1999 on NP bracketing, by Erik Tjong Kim Sang and Miles Osborne. With the help of others, Erik continued the organization of shared tasks until 2003 (on syntactic chunking, clause identification, and named-entity recognition), after which Lluís Màrquez and Xavier Carreras organized two consecutive shared tasks on semantic role labeling (2004, 2005). This year's shared task on multi-lingual dependency parsing holds great promise in becoming a new landmark in NLP research.

With great gratitude we salute all past CoNLL programme chairs and reviewers who have made CoNLL possible, and who have contributed to this conference series, which we believe has a shining future ahead. We are still exploring unknown territory in the fields of language learning, where models of human learning and natural language processing may on one day be one. We hope we will see a long series of CoNLLs along that path.

1997 - Madrid, Spain (chair: T. Mark Ellison)
1998 - Sydney, Australia (chair: David Powers)
1999 - Bergen, Norway (chairs: Miles Osborne and Erik Tjong Kim Sang)
2000 - Lisbon, Portugal (chairs: Claire Cardie, Walter Daelemans, and Erik Tjong Kim Sang)
2001 - Toulouse, France (chairs: Walter Daelemans and Rémi Zajac)
2002 - Taipei, Taiwan (chairs: Dan Roth and Antal van den Bosch)
2003 - Edmonton, Canada (chairs: Walter Daelemans and Miles Osborne)
2004 - Boston, MA, USA (chairs: Hwee Tou Ng and Ellen Riloff)
2005 - Ann Arbor, MI, USA (chairs: Ido Dagan and Dan Gildea)
2006 - New York City, NY, USA (chairs: Lluís Màrquez and Dan Klein)

Antal van den Bosch, President
Hwee Tou Ng, Secretary

Preface

The 2006 Conference on Computational Natural Language Learning is the tenth in a series of yearly meetings organized by SIGNLL, the ACL special interest group on natural language learning. Due to the special occasion, we have brought out the celebratory Roman numerals: welcome to CoNLL-X! Presumably, next year we will return to CoNLL-2007 (until 2016, when perhaps we will see CoNLL-XX). CoNLL-X will be held in New York City on June 8-9, in conjunction with the HLT-NAACL 2006 conference.

A total of 52 papers were submitted to CoNLL's main session, from which only 18 were accepted. The 35% acceptance ratio maintains the high competitiveness of recent CoNLLs and is an indicator of this year's high-quality programme. We are very grateful to the CoNLL community for the large amount of exciting, diverse, and high-quality submissions we received. We are equally grateful to the program committee for their service in reviewing these submissions, on a very tight schedule. Your efforts made our job a pleasure.

As in previous years, we defined a topic of special interest for the conference. This year, we particularly encouraged submissions describing architectures, algorithms, methods, or models designed to improve the robustness of learning-based NLP systems. While the topic of interest was directly addressed by only a small number of the main session submissions, the shared task setting contributed significantly in this direction.

Also following CoNLL tradition, a centerpiece of the confernence is a shared task, this year on multilingual dependency parsing. The shared task was organized by Sabine Buchholz, Amit Dubey, Yuval Krymolwski, and Erwin Marsi, who worked very hard to make the shared task the success it has been. Up to 13 different languages were treated. 19 teams submitted results, from which 17 are presenting description papers in the proceedings. In our opinion, the current shared task constitutes a qualitative step ahead in the evolution of CoNLL shared tasks, and we hope that the resources created and the body of work presented will both serve as a benchmark and also have a substantial impact on future research on syntactic parsing.

Finally, we are delighted to announce that this year's invited speakers are Michael Collins and Walter Daelemans. In accordance with the tenth anniversary celebration, Walter Daelemans will look back at the 10 years of CoNLL conferences, presenting the state of the art in computational natural language learning, and suggesting a new "mission" for the future of field. Michael Collins, in turn, will talk about one of the important current research lines in the field: global learning architectures for structural and relational learning problems in natural language.

In addition to the program committee and shared task organizers, we are very indebted to the SIGNLL board members for very helpful discussion and advice, Erik Tjong Kim Sang, who acted as the information officer, and the HLT-NAACL 2006 conference organizers, in particular Robert Moore, Brian Roark, Sanjeev Khudanpur, Lucy Vanderwende, Roberto Pieraccini, and Liz Liddy for their help with local arrangements and the publication of the proceedings.

To all the attendees, enjoy the CoNLL-X conference!

Lluís Màrquez and Dan Klein
CoNLL-X Program Co-Chairs

Organizers:

Lluís Màrquez, Technical University of Catalonia, Spain
Dan Klein, University of California at Berkeley, USA

Shared Task Organizers:

Sabine Buchholz, Toshiba Research Europe Ltd, UK
Amit Dubey, University of Edinburgh, UK
Yuval Krymolowski, University of Haifa, Israel
Erwin Marsi, Tilburg University, The Netherlands

Information Officer:

Erik Tjong Kim Sang, University of Amsterdam, The Netherlands

Program Committee:

Eneko Agirre, University of the Basque Country, Spain
Regina Barzilay, Massachusetts Institute of Technology, USA
Thorsten Brants, Google Inc., USA
Xavier Carreras, Polytechnical University of Catalunya, Spain
Eugene Charniak, Brown University, USA
Alexander Clark, Royal Holloway University of London, UK
James Cussens, University of York, UK
Walter Daelemans, University of Antwerp, Belgium
Hal Daum, ISI, University of Southern California, USA
Radu Florian, IBM, USA
Dayne Freitag, Fair Isaac Corporation, USA
Daniel Gildea, University of Rochester, USA
Trond Grenager, Stanford University, USA
Marti Hearst, I-School, UC Berkeley, USA
Philipp Koehn, University of Edinburgh, UK
Roger Levy, University of Edinburgh, UK
Rob Malouf, San Diego State University, USA
Christopher Manning, Stanford University, USA
Yuji Matsumoto, Nara Institute of Science and Technology, Japan
Andrew McCallum, University of Massachusetts Amherst, USA
Rada Mihalcea, University of North Texas, USA
Alessandro Moschitti, University of Rome Tor Vergata, Italy
John Nerbonne, University of Groningen, The Netherlands
Hwee Tou Ng, National University of Singapore, Singapore
Franz Josef Och, Google Inc., USA
Miles Osborne, University of Edinburgh, UK

David Powers, Flinders University, Australia
Ellen Riloff, University of Utah, USA
Dan Roth, University of Illinois at Urbana-Champaign, USA
Anoop Sarkar, Simon Fraser University, Canada
Noah Smith, Johns Hopkins University, USA
Suzanne Stevenson, University of Toronto, Canada
Mihai Surdeanu, Polytechnical University of Catalunya, Spain
Charles Sutton, University of Massachusetts Amherst, USA
Kristina Toutanova, Microsoft Research, USA
Antal van den Bosch, Tilburg University, The Netherlands
Janyce Wiebe, University of Pittsburgh, USA
Dekai Wu, Hong Kong University of Science and Technology, Hong Kong

Additional Reviewers:

Sander Canisius, Michael Connor, Andras Csomai, Aron Culotta, Quang Do, Gholamreza Haffari, Yudong Liu, David Martinez, Vanessa Murdoch, Vasin Punyakanok, Lev Ravitov, Kevin Small,Dong Song, Adam Vogel

Invited Speakers:

Michael Collins, Massachusetts Institute of Technology, USA
Walter Daelemans, University of Antwerp, Belgium

Table of Contents

Conference Program

Thursday, June 8, 2006

8:45–8:50 Welcome

Session 1: Syntax and Statistical Parsing

8:50–9:15 *Porting Statistical Parsers with Data-Defined Kernels*
Ivan Titov and James Henderson

9:15–9:40 *Non-Local Modeling with a Mixture of PCFGs*
Slav Petrov, Leon Barrett and Dan Klein

9:40–10:05 *Improved Large Margin Dependency Parsing via Local Constraints and Laplacian Regularization*
Qin Iris Wang, Colin Cherry, Dan Lizotte and Dale Schuurmans

10:05–10:30 *What are the Productive Units of Natural Language Grammar? A DOP Approach to the Automatic Identification of Constructions.*
Willem Zuidema

10:30–11:00 coffee break

11:00–11:50 Invited Talk by Michael Collins

Session 2: Anaphora Resolution and Paraphrasing

11:50–12:15 *Resolving and Generating Definite Anaphora by Modeling Hypernymy using Unlabeled Corpora*
Nikesh Garera and David Yarowsky

12:15–12:40 *Investigating Lexical Substitution Scoring for Subtitle Generation*
Oren Glickman, Ido Dagan, Walter Daelemans, Mikaela Keller and Samy Bengio

12:40–14:00 lunch

Session 3: Shared Task on Dependency Parsing

14:00–15:30 Introduction and System presentation I

15:30–16:00 coffee break

16:00–18:00 System presentation II and Discussion

Friday, June 9, 2006

Session 4: Semantic Role Labeling and Semantics

8:50–9:15 *Semantic Role Recognition Using Kernels on Weighted Marked Ordered Labeled Trees*
Jun'ichi Kazama and Kentaro Torisawa

9:15–9:40 *Semantic Role Labeling via Tree Kernel Joint Inference*
Alessandro Moschitti, Daniele Pighin and Roberto Basili

9:40–10:05 *Can Human Verb Associations Help Identify Salient Features for Semantic Verb Classification?*
Sabine Schulte im Walde

10:05–10:30 *Applying Alternating Structure Optimization to Word Sense Disambiguation*
Rie Kubota Ando

10:30–11:00 coffee break

11:00–11:50 Invited Talk by Walter Daelemans

Session 5: Syntax and Unsupervised Learning

11:50–12:15 *Unsupervised Parsing with U-DOP*
Rens Bod

12:15–12:40 *A Lattice-Based Framework for Enhancing Statistical Parsers with Information from Unlabeled Corpora*
Michaela Atterer and Hinrich Schütze

12:40–14:00 lunch

13:30–14:00 SIGNLL business meeting

Session 6: Thematic Segmentation and Discourse Analysis

14:00–14:25 *Word Distributions for Thematic Segmentation in a Support Vector Machine Approach*
Maria Georgescul, Alexander Clark and Susan Armstrong

14:25–14:50 *Which Side are You on? Identifying Perspectives at the Document and Sentence Levels*
Wei-Hao Lin, Theresa Wilson, Janyce Wiebe and Alexander Hauptmann

Friday, June 9, 2006 (continued)

Session 7: Grammatical Inference

14:50–15:15 *Unsupervised Grammar Induction by Distribution and Attachment*
David J. Brooks

15:15–15:40 *Learning Auxiliary Fronting with Grammatical Inference*
Alexander Clark and Rémi Eyraud

15:40–16:00 coffee break

Session 8: Information Extraction and Named Entity Extraction

16:00–16:25 *Using Gazetteers in Discriminative Information Extraction*
Andrew Smith and Miles Osborne

16:25–16:50 *A Context Pattern Induction Method for Named Entity Extraction*
Partha Pratim Talukdar, Thorsten Brants, Mark Liberman and Fernando Pereira

16:50–17:00 Best Paper Award

17:00 Closing

A Mission for Computational Natural Language Learning

Walter Daelemans
CNTS Language Technology Group
University of Antwerp
Belgium
walter.daelemans@ua.ac.be

Abstract

In this presentation, I will look back at 10 years of CoNLL conferences and the state of the art of machine learning of language that is evident from this decade of research. My conclusion, intended to provoke discussion, will be that we currently lack a clear motivation or "mission" to survive as a discipline. I will suggest that a new mission for the field could be found in a renewed interest for theoretical work (which learning algorithms have a bias that matches the properties of language?, what is the psycholinguistic relevance of learner design issues?), in more sophisticated comparative methodology, and in solving the problem of transfer, reusability, and adaptation of learned knowledge.

1 Introduction

When looking at ten years of CoNLL conferences, it is clear that the impact and the size of the conference has enormously grown over time. The technical papers you will find in this proceedings now are comparable in quality and impact to those of other distinguished conferences like the *Conference on Empirical Methods in Natural Language Processing* or even the main conferences of ACL, EACL and NAACL themselves. An important factor in the success of CoNLL has been the continued series of shared tasks (notice we don't use terms like *challenges* or *competitions*) that has produced a useful set of benchmarks for comparing learning methods, and that has gained wide interest in the field. It should also be noted, however, that the success of the conferences is inversely proportional with the degree to which the original topics which motivated the conference are present in the programme. Originally, the people driving CoNLL wanted it to be promiscuous (i) in the selection of partners (we wanted to associate with Machine Learning, Linguistics and Cognitive Science conferences as well as with Computational Linguistics conferences) and (ii) in the range of topics to be presented. We wanted to encourage linguistically and psycholinguistically relevant machine learning work, and biologically inspired and innovative symbolic learning methods, and present this work alongside the statistical and learning approaches that were at that time only starting to gradually become the mainstream in Computational Linguistics. It has turned out differently, and we should reflect on whether we have become too much of a mainstream computational linguistics conference ourselves, a back-off for the good papers that haven't made it in EMNLP or ACL because of the crazy rejection rates there (with EMNLP in its turn a back-off for good papers that haven't made it in ACL). Some of the work targeted by CoNLL has found a forum in meetings like the workshop on *Psycho-computational models of human language acquisition*, the *International Colloquium on Grammatical Inference*, the workshop on *Morphological and Phonological Learning* etc. We should ask ourselves why we don't have this type of work more in CoNLL. In the first part of the presentation I will sketch *very* briefly the history of SIGNLL and

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 1–5, New York City, June 2006.

CoNLL and try to initiate some discussion on what a conference on Computational Language Learning should be doing in 2007 and after.

2 State of the Art in Computational Natural Language Learning

The second part of my presentation will be a discussion of the state of the art as it can be found in CoNLL (and EMNLP and the ACL conferences). The field can be divided into theoretical, methodological, and engineering work. There has been progress in theory and methodology, but perhaps not sufficiently. I will argue that most progress has been made in *engineering* with most often incremental progress on specific tasks as a result rather than increased understanding of how language can be learned from data.

Machine Learning of Natural Language (MLNL), or Computational Natural Language Learning (CoNLL) is a research area lying in the intersection of computational linguistics and machine learning. I would suggest that Statistical Natural Language Processing (SNLP) should be treated as part of MLNL, or perhaps even as a synonym. Symbolic machine learning methods belong to the same part of the ontology as statistical methods, but have different solutions for specific problems. E.g., Inductive Logic Programming allows elegant addition of background knowledge, memory-based learning has implicit similarity-based smoothing, etc.

There is no need here to explain the success of inductive methods in Computational Linguistics and why we are all such avid users of the technology: availability of data, fast production of systems with good accuracy, robustness and coverage, cheaper than linguistic labor. There is also no need here to explain that many of these arguments in favor of learning in NLP are bogus. Getting statistical and machine learning systems to work involves design, optimization, and smoothing issues that are something of a black art. For many problems, getting sufficient annotated data is expensive and difficult, our annotators don't sufficiently agree, our trained systems are not really that good. My favorite example for the latter is part of speech tagging, which is considered a solved problem, but still has error rates of 20-30% for the ambiguities that count, like verb-noun ambiguity. We are doing better than handcrafted linguistic knowledge-based approaches but from the point of view of the goal of robust language understanding unfortunately not that significantly better. Twice better than very bad is not necessarily any good. We also implicitly redefined the goals of the field of Computational Linguistics, forgetting for example about quantification, modality, tense, inference and a large number of other sentence and discourse semantics issues which do not fit the default classification-based supervised learning framework very well or for which we don't have annotated data readily available. As a final irony, one of the reasons why learning methods have become so prevalent in NLP is their *success* in speech recognition. Yet, there too, this success is relative; the goal of spontaneous speaker-independent recognition is still far away.

2.1 Theory

There has been a lot of progress recently in theoretical machine learning(Vapnik, 1995; Jordan, 1999). Statistical Learning Theory and progress in Graphical Models theory have provided us with a well-defined framework in which we can relate different approaches like kernel methods, Naive Bayes, Markov models, maximum entropy approaches (logistic regression), perceptrons and CRFs. Insight into the differences between generative and discriminative learning approaches has clarified the relations between different learning algorithms considerably.

However, this work does not tell us something general about machine learning *of language*. Theoretical issues that should be studied in MLNL are for example which classes of learning algorithms are best suited for which type of language processing task, what the need for training data is for a given task, which information sources are necessary and sufficient for learning a particular language processing task, etc. These fundamental questions all relate to learning algorithm *bias* issues. Learning is a search process in a hypothesis space. Heuristic limitations on the search process and restrictions on the representations allowed for input and hypothesis representations together define this bias. There is not a lot of work on matching properties of learning algorithms with properties of language processing

tasks, or more specifically on how the bias of particular (families of) learning algorithms relates to the hypothesis spaces of particular (types of) language processing tasks.

As an example of such a unifying approach, (Roth, 2000) shows that several different algorithms (memory-based learning, tbl, snow, decision lists, various statistical learners, ...) use the same type of knowledge representation, a linear representation over a feature space based on a transformation of the original instance space. However, the only relation to language here is rather negative with the claim that this bias is not sufficient for learning higher level language processing tasks.

As another example of this type of work, Memory-Based Learning (MBL) (Daelemans and van den Bosch, 2005), with its implicit similarity-based smoothing, storage of all training evidence, and uniform modeling of regularities, subregularities and exceptions has been proposed as having the right bias for language processing tasks. Language processing tasks are mostly governed by Zipfian distributions and high disjunctivity which makes it difficult to make a principled distinction between noise and exceptions, which would put *eager* learning methods (i.e. most learning methods apart from MBL and kernel methods) at a disadvantage.

More theoretical work in this area should make it possible to relate machine learner bias to properties of language processing tasks in a more fine-grained way, providing more insight into both language and learning. An avenue that has remained largely unexplored in this respect is the use of artificial data emulating properties of language processing tasks, making possible a much more fine-grained study of the influence of learner bias. However, research in this area will not be able to ignore the "no free lunch" theorem (Wolpert and Macready, 1995). Referring back to the problem of induction (Hume, 1710) this theorem can be interpreted that no inductive algorithm is universally better than any other; generalization performance of any inductive algorithm is zero when averaged over a uniform distribution of all possible classification problems (i.e. assuming a random universe). This means that the only way to test hypotheses about bias and necessary information sources in language learning is to perform empirical research, making a reliable experimental methodology necessary.

2.2 Methodology

Either to investigate the role of different information sources in learning a task, or to investigate whether the bias of some learning algorithm fits the properties of natural language processing tasks better than alternative learning algorithms, *comparative* experiments are necessary. As an example of the latter, we may be interested in investigating whether part-of-speech tagging improves the accuracy of a Bayesian text classification system or not. As an example of the former, we may be interested to know whether a relational learner is better suited than a propositional learner to learn semantic function association. This can be achieved by comparing the accuracy of the learner with and without the information source or different learners on the same task. Crucial for objectively comparing algorithm bias and relevance of information sources is a methodology to reliably measure differences and compute their statistical significance. A detailed methodology has been developed for this involving approaches like k-fold cross-validation to estimate classifier quality (in terms of measures derived from a confusion matrix like accuracy, precision, recall, F-score, ROC, AUC, etc.), as well as statistical techniques like McNemar and paired cross-validation t-tests for determining the statistical significance of differences between algorithms or between presence or absence of information sources. This methodology is generally accepted and used both in machine learning and in most work in inductive NLP.

CoNLL has contributed a lot to this comparative work by producing a successful series of shared tasks, which has provided to the community a rich set of benchmark language processing tasks. Other competitive research evaluations like senseval, the PASCAL challenges and the NIST competitions have similarly tuned the field toward comparative learning experiments. In a typical comparative machine learning experiment, two or more algorithms are compared for a fixed sample selection, feature selection, feature representation, and (default) algorithm parameter setting over a number of trials (cross-validation), and if the measured differences are statistically significant, conclusions are drawn about which algorithm is better suited to the problem

being studied and why (mostly in terms of algorithm bias). Sometimes different sample sizes are used to provide a learning curve, and sometimes parameters of (some of the) algorithms are optimized on training data, or heuristic feature selection is attempted, but this is exceptional rather than common practice in comparative experiments.

Yet everyone knows that many factors potentially play a role in the outcome of a (comparative) machine learning experiment: the data used (the sample selection and the sample size), the information sources used (the features selected) and their representation (e.g. as nominal or binary features), the class representation (error coding, binarization of classes), and the algorithm parameter settings (most ML algorithms have various parameters that can be tuned). Moreover,all these factors are known to interact. E.g., (Banko and Brill, 2001) demonstrated that for confusion set disambiguation, a prototypical disambiguation in context problem, the amount of data used dominates the effect of the bias of the learning method employed. The effect of training data size on relevance of POS-tag information on top of lexical information in relation finding was studied in (van den Bosch and Buchholz, 2001). The positive effect of POS-tags disappears with sufficient data. In (Daelemans et al., 2003) it is shown that the joined optimization of feature selection and algorithm parameter optimization significantly improves accuracy compared to sequential optimization. Results from comparative experiments may therefore not be reliable. I will suggest an approach to improve methodology to improve reliability.

2.3 Engineering

Whereas comparative machine learning work can potentially provide useful theoretical insights and results, there is a distinct feeling that it also leads to an exaggerated attention for accuracy *on the dataset*. Given the limited transfer and reusability of learned modules when used in different domains, corpora etc., this may not be very relevant. If a WSJ-trained statistical parser looses 20% accuracy on a comparable newspaper testcorpus, it doesn't really matter a lot that system A does 1% better than system B on the default WSJ-corpus partition.

In order to win shared tasks and perform best on some language processing task, various clever architectural and algorithmic variations have been proposed, sometimes with the single goal of getting higher accuracy (ensemble methods, classifier combination in general, ...), sometimes with the goal of solving manual annotation bottlenecks (active learning, co-training, semisupervised methods, ...).

This work is extremely valid from the point of view of computational linguistics researchers looking for any old method that can boost performance and get benchmark natural language processing problems or applications solved. But from the point of view of a SIG on computational natural language learning, this work is probably too much theory-independent and doesn't teach us enough about *language* learning.

However, engineering work like this can suddenly become theoretically important when motivated not by a few percentage decimals more accuracy but rather by (psycho)linguistic plausibility. For example, the current trend in combining local classifiers with holistic inference may be a cognitively relevant principle rather than a neat engineering trick.

3 Conclusion

The field of computational natural language learning is in need of a renewed mission. In two parent fields dominated by good engineering use of machine learning in language processing, and interesting developments in computational language learning respectively, our field should focus more on theory. More research should address the question what we can learn about language from comparative machine learning experiments, and address or at least acknowledge methodological problems.

4 Acknowledgements

There are many people that have influenced me, most of my students and colleagues have done so at some point, but I would like to single out David Powers and Antal van den Bosch, and thank them for making this strange field of computational language learning such an interesting and pleasant playground.

References

Michele Banko and Eric Brill. 2001. Mitigating the paucity-of-data problem: exploring the effect of train-

ing corpus size on classifier performance for natural language processing. In *HLT '01: Proceedings of the first international conference on Human language technology research*, pages 1–5, Morristown, NJ, USA. Association for Computational Linguistics.

Walter Daelemans and Antal van den Bosch. 2005. *Memory-Based Language Processing*. Cambridge University Press, Cambridge, UK.

Walter Daelemans, Véronique Hoste, Fien De Meulder, and Bart Naudts. 2003. Combined optimization of feature selection and algorithm parameter interaction in machine learning of language. In *Proceedings of the 14th European Conference on Machine Learning (ECML-2003)*, Lecture Notes in Computer Science 2837, pages 84–95, Cavtat-Dubrovnik, Croatia. Springer-Verlag.

D. Hume. 1710. *A Treatise Concerning the Principles of Human Knowledge*.

M. I. Jordan. 1999. *Learning in graphical models*. MIT, Cambridge, MA, USA.

D. Roth. 2000. Learning in natural language: Theory and algorithmic approaches. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–6, Lisbon, Portugal.

Antal van den Bosch and Sabine Buchholz. 2001. Shallow parsing on the basis of words only: a case study. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 433–440, Morristown, NJ, USA. Association for Computational Linguistics.

Vladimir N. Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.

David H. Wolpert and William G. Macready. 1995. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe, NM.

Porting Statistical Parsers with Data-Defined Kernels

Ivan Titov
University of Geneva
24, rue Général Dufour
CH-1211 Genève 4, Switzerland
ivan.titov@cui.unige.ch

James Henderson
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW, United Kingdom
james.henderson@ed.ac.uk

Abstract

Previous results have shown disappointing performance when porting a parser trained on one domain to another domain where only a small amount of data is available. We propose the use of data-defined kernels as a way to exploit statistics from a source domain while still specializing a parser to a target domain. A probabilistic model trained on the source domain (and possibly also the target domain) is used to define a kernel, which is then used in a large margin classifier trained only on the target domain. With a SVM classifier and a neural network probabilistic model, this method achieves improved performance over the probabilistic model alone.

1 Introduction

In recent years, significant progress has been made in the area of natural language parsing. This research has focused mostly on the development of statistical parsers trained on large annotated corpora, in particular the Penn Treebank WSJ corpus (Marcus et al., 1993). The best statistical parsers have shown good results on this benchmark, but these statistical parsers demonstrate far worse results when they are applied to data from a different domain (Roark and Bacchiani, 2003; Gildea, 2001; Ratnaparkhi, 1999). This is an important problem because we cannot expect to have large annotated corpora available for most domains. While identifying this problem, previous work has not proposed parsing methods which are specifically designed for porting parsers. Instead they propose methods for training a standard parser with a large amount of out-of-domain data and a small amount of in-domain data.

In this paper, we propose using data-defined kernels and large margin methods to specifically address porting a parser to a new domain. Data-defined kernels are used to construct a new parser which exploits information from a parser trained on a large out-of-domain corpus. Large margin methods are used to train this parser to optimize performance on a small in-domain corpus.

Large margin methods have demonstrated substantial success in applications to many machine learning problems, because they optimize a measure which is directly related to the expected testing performance. They achieve especially good performance compared to other classifiers when only a small amount of training data is available. Most of the large margin methods need the definition of a kernel. Work on kernels for natural language parsing has been mostly focused on the definition of kernels over parse trees (e.g. (Collins and Duffy, 2002)), which are chosen on the basis of domain knowledge. In (Henderson and Titov, 2005) it was proposed to apply a class of kernels derived from probabilistic models to the natural language parsing problem.

In (Henderson and Titov, 2005), the kernel is constructed using the parameters of a trained probabilistic model. This type of kernel is called a data-defined kernel, because the kernel incorporates information from the data used to train the probabilistic model. We propose to exploit this property to transfer information from a large corpus to a statis-

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 6–13, New York City, June 2006.

tical parser for a different domain. Specifically, we propose to train a statistical parser on data including the large corpus, and to derive the kernel from this trained model. Then this derived kernel is used in a large margin classifier trained on the small amount of training data available for the target domain.

In our experiments, we consider two different scenarios for porting parsers. The first scenario is the pure porting case, which we call "transferring". Here we only require a probabilistic model trained on the large corpus. This model is then reparameterized so as to extend the vocabulary to better suit the target domain. The kernel is derived from this reparameterized model. The second scenario is a mixture of parser training and porting, which we call "focusing". Here we train a probabilistic model on both the large corpus and the target corpus. The kernel is derived from this trained model. In both scenarios, the kernel is used in a SVM classifier (Tsochantaridis et al., 2004) trained on a small amount of data from the target domain. This classifier is trained to rerank the candidate parses selected by the associated probabilistic model. We use the Penn Treebank Wall Street Journal corpus as the large corpus and individual sections of the Brown corpus as the target corpora (Marcus et al., 1993). The probabilistic model is a neural network statistical parser (Henderson, 2003), and the data-defined kernel is a TOP reranking kernel (Henderson and Titov, 2005).

With both scenarios, the resulting parser demonstrates improved accuracy on the target domain over the probabilistic model alone. In additional experiments, we evaluate the hypothesis that the primary issue for porting parsers between domains is differences in the distributions of words in structures, and not in the distributions of the structures themselves. We partition the parameters of the probability model into those which define the distributions of words and those that only involve structural decisions, and derive separate kernels for these two subsets of parameters. The former model achieves virtually identical accuracy to the full model, but the later model does worse, confirming the hypothesis.

2 Data-Defined Kernels for Parsing

Previous work has shown how data-defined kernels can be applied to the parsing task (Henderson and Titov, 2005). Given the trained parameters of a probabilistic model of parsing, the method defines a kernel over sentence-tree pairs, which is then used to rerank a list of candidate parses.

In this paper, we focus on the TOP reranking kernel defined in (Henderson and Titov, 2005), which are closely related to Fisher kernels. The reranking task is defined as selecting a parse tree from the list of candidate trees $(y_1, \ldots, y_s)$ suggested by a probabilistic model $P(x, y|\hat{\theta})$, where $\hat{\theta}$ is a vector of model parameters learned during training the probabilistic model. The motivation for the TOP reranking kernel is given in (Henderson and Titov, 2005), but for completeness we note that the its feature extractor is given by:

$$\phi_{\hat{\theta}}(x, y_k) = \\ (v(x, y_k, \hat{\theta}), \tfrac{\partial v(x,y_k,\hat{\theta})}{\partial \theta_1}, \ldots, \tfrac{\partial v(x,y_k,\hat{\theta})}{\partial \theta_l}), \quad (1)$$

where $v(x, y_k, \hat{\theta}) = \log P(x, y_k|\hat{\theta}) - \log \sum_{t \neq k} P(x, y_t|\hat{\theta})$. The first feature reflects the score given to (x, y_k) by the probabilistic model (relative to the other candidates for x), and the remaining features reflect how changing the parameters of the probabilistic model would change this score for (x, y_k).

The parameters $\hat{\theta}$ used in this feature extractor do not have to be exactly the same as the parameters trained in the probabilistic model. In general, we can first reparameterize the probabilistic model, producing a new model which defines exactly the same probability distribution as the old model, but with a different set of adjustable parameters. For example, we may want to freeze the values of some parameters (thereby removing them from $\hat{\theta}$), or split some parameters into multiple cases (thereby duplicating their values in $\hat{\theta}$). This flexibility allows the features used in the kernel method to be different from those used in training the probabilistic model. This can be useful for computational reasons, or when the kernel method is not solving exactly the same problem as the probabilistic model was trained for.

3 Porting with Data-Defined Kernels

In this paper, we consider porting a parser trained on a large amount of annotated data to a different domain where only a small amount of annotated data is available. We validate our method in two different

scenarios, transferring and focusing. Also we verify the hypothesis that addressing differences between the vocabularies of domains is more important than addressing differences between their syntactic structures.

3.1 Transferring to a Different Domain

In the transferring scenario, we are given just a probabilistic model which has been trained on a large corpus from a source domain. The large corpus is not available during porting, and the small corpus for the target domain is not available during training of the probabilistic model. This is the case of pure parser porting, because it only requires the source domain parser, not the source domain corpus. Besides this theoretical significance, this scenario has the advantage that we only need to train a single probabilistic parser, thereby saving on training time and removing the need for access to the large corpus once this training is done. Then any number of parsers for new domains can be trained, using only the small amount of annotated data available for the new domain.

Our proposed porting method first constructs a data-defined kernel using the parameters of the trained probabilistic model. A large margin classifier with this kernel is then trained to rerank the top candidate parses produced by the probabilistic model. Only the small target corpus is used during training of this classifier. The resulting parser consists of the original parser plus a very computationally cheap procedure to rerank its best parses.

Whereas training of standard large margin methods, like SVMs, isn't feasible on a large corpus, it is quite tractable to train them on a small target corpus.[1] Also, the choice of the large margin classifier is motivated by their good generalization properties on small datasets, on which accurate probabilistic models are usually difficult to learn.

We hypothesize that differences in vocabulary across domains is one of the main difficulties with parser portability. To address this problem, we propose constructing the kernel from a probabilistic model which has been reparameterized to better suit the target domain vocabulary. As in other lexicalized statistical parsers, the probabilistic model we use treats words which are not frequent enough in the training set as 'unknown' words (Henderson, 2003). Thus there are no parameters in this model which are specifically for these words. When we consider a different target domain, a substantial proportion of the words in the target domain are treated as unknown words, which makes the parser only weakly lexicalized for this domain.

To address this problem, we reparameterize the probability model so as to add specific parameters for the words which have high enough frequency in the target domain training set but are treated as unknown words by the original probabilistic model. These new parameters all have the same values as their associated unknown words, so the probability distribution specified by the model does not change. However, when a kernel is defined with this reparameterized model, the kernel's feature extractor includes features specific to these words, so the training of a large margin classifier can exploit differences between these words in the target domain. Expanding the vocabulary in this way is also justified for computational reasons; the speed of the probabilistic model we use is greatly effected by vocabulary size, but the large-margin method is not.

3.2 Focusing on a Subdomain

In the focusing scenario, we are given the large corpus from the source domain. We may also be given a parsing model, but as with other approaches to this problem we simply throw this parsing model away and train a new one on the combination of the source and target domain data. Previous work (Roark and Bacchiani, 2003) has shown that better accuracy can be achieved by finding the optimal re-weighting between these two datasets, but this issue is orthogonal to our method, so we only consider equal weighting. After this training phase, we still want to optimize the parser for only the target domain.

Once we have the trained parsing model, our proposed porting method proceeds the same way in this scenario as in transferring. However, because the original training set already includes the vocabulary from the target domain, the reparameterization approach defined in the preceding section is not necessary so we do not perform it. This reparameter-

[1] In (Shen and Joshi, 2003) it was proposed to use an ensemble of SVMs trained the Wall Street Journal corpus, but we believe that the generalization performance of the resulting classifier is compromised in this approach.

ization could be applied here, thereby allowing us to use a statistical parser with a smaller vocabulary, which can be more computationally efficient both during training and testing. However, we would expect better accuracy of the combined system if the same large vocabulary is used both by the probabilistic parser and the kernel method.

3.3 Vocabulary versus Structure

It is commonly believed that differences in vocabulary distributions between domains effects the ported parser performance more significantly than the differences in syntactic structure distributions. We would like to test this hypothesis in our framework. The probabilistic model (Henderson, 2003) allows us to distinguish between those parameters responsible for the distributions of individual vocabulary items, and those parameters responsible for the distributions of structural decisions, as described in more details in section 4.2. We train two additional models, one which uses a kernel defined in terms of only vocabulary parameters, and one which uses a kernel defined in terms of only structure parameters. By comparing the performance of these models and the model with the combined kernel, we can draw conclusion on the relative importance of vocabulary and syntactic structures for parser portability.

4 An Application to a Neural Network Statistical Parser

Data-defined kernels can be applied to any kind of parameterized probabilistic model, but they are particularly interesting for latent variable models. Without latent variables (e.g. for PCFG models), the features of the data-defined kernel (except for the first feature) are a function of the counts used to estimate the model. For a PCFG, each such feature is a function of one rule's counts, where the counts from different candidates are weighted using the probability estimates from the model. With latent variables, the meaning of the variable (not just its value) is learned from the data, and the associated features of the data-defined kernel capture this induced meaning. There has been much recent work on latent variable models (e.g. (Matsuzaki et al., 2005; Koo and Collins, 2005)). We choose to use an earlier neural network based probabilistic model of parsing (Henderson, 2003), whose hidden units can be viewed as approximations to latent variables. This parsing model is also a good candidate for our experiments because it achieves state-of-the-art results on the standard Wall Street Journal (WSJ) parsing problem (Henderson, 2003), and data-defined kernels derived from this parsing model have recently been used with the Voted Perceptron algorithm on the WSJ parsing task, achieving a significant improvement in accuracy over the neural network parser alone (Henderson and Titov, 2005).

4.1 The Probabilistic Model of Parsing

The probabilistic model of parsing in (Henderson, 2003) has two levels of parameterization. The first level of parameterization is in terms of a history-based generative probability model. These parameters are estimated using a neural network, the weights of which form the second level of parameterization. This approach allows the probability model to have an infinite number of parameters; the neural network only estimates the bounded number of parameters which are relevant to a given partial parse. We define our kernels in terms of the second level of parameterization (the network weights).

A history-based model of parsing first defines a one-to-one mapping from parse trees to sequences of parser decisions, $d_1,...,d_m$ (i.e. derivations). Henderson (2003) uses a form of left-corner parsing strategy, and the decisions include generating the words of the sentence (i.e. it is generative). The probability of a sequence $P(d_1,...,d_m)$ is then decomposed into the multiplication of the probabilities of each parser decision conditioned on its history of previous decisions $\Pi_i P(d_i|d_1,...,d_{i-1})$.

4.2 Deriving the Kernel

The complete set of neural network weights isn't used to define the kernel, but instead reparameterization is applied to define a third level of parameterization which only includes the network's output layer weights. As suggested in (Henderson and Titov, 2005) use of the complete set of weights doesn't lead to any improvement of the resulting reranker and makes the reranker training more computationally expensive.

Furthermore, to assess the contribution of vocabulary and syntactic structure differences (see sec-

tion 3.3), we divide the set of the parameters into vocabulary parameters and structural parameters. We consider the parameters used in the estimation of the probability of the next word given the history representation as vocabulary parameters, and the parameters used in the estimation of structural decision probabilities as structural parameters. We define the kernel with structural features as using only structural parameters, and the kernel with vocabulary features as using only vocabulary parameters.

5 Experimental Results

We used the Penn Treebank WSJ corpus and the Brown corpus to evaluate our approach. We used the standard division of the WSJ corpus into training, validation, and testing sets. In the Brown corpus we ran separate experiments for sections F (informative prose: popular lore), K (imaginative prose: general fiction), N (imaginative prose: adventure and western fiction), and P (imaginative prose: romance and love story). These sections were selected because they are sufficiently large, and because they appeared to be maximally different from each other and from WSJ text. In each Brown corpus section, we selected every third sentence for testing. From the remaining sentences, we used 1 sentence out of 20 for the validation set, and the remainder for training. The resulting datasets sizes are presented in table 1.

For the large margin classifier, we used the SVM-Struct (Tsochantaridis et al., 2004) implementation of SVM, which rescales the margin with F_1 measure of bracketed constituents (see (Tsochantaridis et al., 2004) for details). Linear slack penalty was employed.[2]

5.1 Experiments on Transferring across Domains

To evaluate the pure porting scenario (transferring), described in section 3.1, we trained the SSN parsing model on the WSJ corpus. For each tag, there is an unknown-word vocabulary item which is used for all those words not sufficiently frequent with that tag to be included individually in the vocabulary. In the vocabulary of the parser, we included the unknown-word items and the words which occurred in the training set at least 20 times. This led to the vocabulary of 4,215 tag-word pairs.

	testing	training	validation
WSJ	2,416 (54,268)	39,832 (910,196)	1,346 (31,507)
Brown F	1,054 (23,722)	2,005 (44,928)	105 (2,300)
Brown K	1,293 (21,215)	2,459 (39,823)	129 (1,971)
Brown N	1,471 (22,142)	2,797 (42,071)	137 (2,025)
Brown P	1,314 (21,763)	2,503 (41,112)	125 (1,943)

Table 1: Number of sentences (words) for each dataset.

We derived the kernel from the trained model for each target section (F, K, N, P) using reparameterization discussed in section 3.1: we included in the vocabulary all the words which occurred at least twice in the training set of the corresponding section. This approach led to a smaller vocabulary than that of the initial parser but specifically tied to the target domain (3,613, 2,789, 2,820 and 2,553 tag-word pairs for sections F, K, N and P respectively). There is no sense in including the words from the WSJ which do not appear in the Brown section training set because the classifier won't be able to learn the corresponding components of its decision vector. The results for the original probabilistic model (SSN-WSJ) and for the kernel method (TOP-Transfer) on the testing set of each section are presented in table 2.[3]

To evaluate the relative contribution of our porting technique versus the use of the TOP kernel alone, we also used this TOP kernel to train an SVM on the WSJ corpus. We trained the SVM on data from the development set and section 0, so that the size of this dataset (3,267 sentences) was about the same as for each Brown section.[4] This gave us a "TOP-WSJ"

[2]Training of the SVM takes about 3 hours on a standard desktop PC. Running the SVM is very fast, once the probabilistic model has finished computing the probabilities needed to select the candidate parses.

[3]All our results are computed with the evalb program following the standard criteria in (Collins, 1999).

[4]We think that using an equivalently sized dataset provides a fair test of the contribution of the TOP kernel alone. It would also not be computationally tractable to train an SVM on the full WSJ dataset without using different training techniques, which would then compromise the comparison.

model, which we tested on each of the four Brown sections. In each case, the TOP-WSJ model did worse than the original SSN-WSJ model, as shown in table 2. This makes it clear that we are getting no improvement from simply using a TOP kernel alone or simply using more data, and all our improvement is from the proposed porting method.

5.2 Experiments on Focusing on a Subdomain

To perform the experiments on the approach suggested in section 3.2 (focusing), we trained the SSN parser on the WSJ training set joined with the training set of the corresponding section. We included in the vocabulary only words which appeared in the joint training set at least 20 times. Resulting vocabularies comprised 4,386, 4,365, 4,367 and 4,348 for sections F, K, N and P, respectively.[5] Experiments were done in the same way as for the parser transferring approach, but reparameterization was not performed. Standard measures of accuracy for the original probabilistic model (SSN-WSJ+Br) and the kernel method (TOP-Focus) are also shown in table 2.

For the sake of comparison, we also trained the SSN parser on only training data from one of the Brown corpus sections (section P), producing a "SSN-Brown" model. This model achieved an F_1 measure of only 81.0% for the P section testing set, which is worse than all the other models and is 3% lower than our best results on this testing set (TOP-Focus). This result underlines the need to port parsers from domains in which there are large annotated datasets.

5.3 Experiments Comparing Vocabulary to Structure

We conducted the same set of experiments with the kernel with vocabulary features (TOP-Voc-Transfer and TOP-Voc-Focus) and with the kernel with the structural features (TOP-Str-Transfer and TOP-Str-Focus). Average results for classifiers with these kernels, as well as for the original kernel and the baseline, are presented in table 3.

[5]We would expect some improvement if we used a smaller threshold on the target domain, but preliminary results suggest that this improvement would be small.

	section	LR	LP	$F_{\beta=1}$
TOP-WSJ	F	83.9	84.9	84.4
SSN-WSJ	F	84.4	85.2	84.8
TOP-Transfer	F	84.5	85.6	85.0
SSN-WSJ+Br	F	84.2	85.2	84.7
TOP-Focus	F	84.6	86.0	85.3
TOP-WSJ	K	81.8	82.3	82.1
SSN-WSJ	K	82.2	82.6	82.4
TOP-Transfer	K	82.4	83.5	83.0
SSN-WSJ+Br	K	83.1	84.2	83.6
TOP-Focus	K	83.6	85.0	84.3
TOP-WSJ	N	83.3	84.5	83.9
SSN-WSJ	N	83.5	84.6	84.1
TOP-Transfer	N	84.3	85.7	85.0
SSN-WSJ+Br	N	85.0	86.5	85.7
TOP-Focus	N	85.0	86.7	85.8
TOP-WSJ	P	81.3	82.1	81.7
SSN-WSJ	P	82.3	83.0	82.6
TOP-Transfer	P	82.7	83.8	83.2
SSN-WSJ+Br	P	83.1	84.3	83.7
TOP-Focus	P	83.3	84.8	84.0

Table 2: Percentage labeled constituent recall (LR), precision (LP), and a combination of both ($F_{\beta=1}$) on the individual test sets.

5.4 Discussion of Results

For the experiments which directly test the usefulness of our proposed porting technique (SSN-WSJ versus TOP-Transfer), our technique demonstrated improvement for each of the Brown sections (table 2), and this improvement was significant for three out of four of the sections (K, N, and P).[6] This demonstrates that data-defined kernels are an effective way to port parsers to a new domain.

For the experiments which combine training a new probability model with our porting technique (SSN-WSJ+Br versus TOP-Focus), our technique still demonstrated improvement over training alone. There was improvement for each of the Brown sections, and this improvement was significant for two

[6]We measured signifi cance in F_1 measure at the 5% level with the randomized signifi cance test of (Yeh, 2000). We think that the reason the improvement on section F was only significant at the 10% level was that the baseline model (SSN-WSJ) was particularly lucky, as indicated by the fact that it did even better than the model trained on the combination of datasets (SSN-WSJ+Br).

	LR	LP	$F_{\beta=1}$
SSN-WSJ	83.1	83.8	83.5
TOP-Transfer	83.5	84.7	84.1
TOP-Voc-Transfer	83.5	84.7	84.1
TOP-Str-Transfer	83.1	84.3	83.7
SSN-WSJ+Br	83.8	85.0	84.4
TOP-Focus	84.1	85.6	84.9
TOP-Voc-Focus	84.1	85.6	84.8
TOP-Str-Focus	83.9	85.4	84.7

Table 3: Average accuracy of the models on chapters F, K, N and P of the Brown corpus.

out of four of the sections (F and K). This demonstrates that, even when the probability model is well suited to the target domain, there is still room for improvement from using data-defined kernels to optimize the parser specifically to the target domain without losing information about the source domain.

One potential criticism of these conclusions is that the improvement could be the result of reranking with the TOP kernel, and have nothing to do with porting. The lack of an improvement in the TOP-WSJ results discussed in section 5.1 clearly shows that this cannot be the explanation. The opposite criticism is that the improvement could be the result of optimizing to the target domain alone. The poor performance of the SSN-Brown model discussed in section 5.2 makes it clear that this also cannot be the explanation. Therefore reranking with data defined kernels must be both effective at preserving information about the source domain and effective at specializing to the target domain.

The experiments which test the hypothesis that differences in vocabulary distributions are more important than difference in syntactic structure distributions confirm this belief. Results for the classifier which uses the kernel with only vocabulary features are better than those for structural features in each of the four sections with both the Transfer and Focus scenarios. In addition, comparing the results of TOP-Transfer with TOP-Voc-Transfer and TOP-Focus with TOP-Voc-Focus, we can see that adding structural features in TOP-Focus and TOP-Transfer leads to virtually no improvement. This suggest that differences in vocabulary distributions are the only issue we need to address, although this result could possibly also be an indication that our method did not sufficiently exploit structural differences.

In this paper we concentrate on the situation where a parser is needed for a restricted target domain, for which only a small amount of data is available. We believe that this is the task which is of greatest practical interest. For this reason we do not run experiments on the task considered in (Gildea, 2001) and (Roark and Bacchiani, 2003), where they are porting from the restricted domain of the WSJ corpus to the more varied domain of the Brown corpus as a whole. However, to help emphasize the success of our proposed porting method, it is relevant to show that even our baseline models are performing better than this previous work on parser portability. We trained and tested the SSN parser in their "de-focusing" scenario using the same datasets as (Roark and Bacchiani, 2003). When trained only on the WSJ data (analogously to the SSN-WSJ baseline for TOP-Transfer) it achieves results of 82.9%/83.4% LR/LP and 83.2% F_1, and when trained on data from both domains (analogously to the SSN-WSJ+Br baselines for TOP-Focus) it achieves results of 86.3%/87.6% LR/LP and 87.0% F_1. These results represent a 2.2% and 1.3% increase in F_1 over the best previous results, respectively (see the discussion of (Roark and Bacchiani, 2003) below).

6 Related Work

Most research in the field of parsing has focused on the Wall Street Journal corpus. Several researchers have addressed the portability of these WSJ parsers to other domains, but mostly without addressing the issue of how a parser can be designed specifically for porting to another domain. Unfortunately, no direct empirical comparison is possible between our results and results with other parsers, because there is no standard portability benchmark to date where a small amount of data from a target domain is used.

(Ratnaparkhi, 1999) performed portability experiments with a Maximum Entropy parser and demonstrated that the parser trained on WSJ achieves far worse results on the Brown corpus sections. Adding a small amount of data from the target domain improves the results, but accuracy is still much lower than the results on the WSJ. They reported results when their parser was trained on the WSJ training

set plus a portion of 2,000 sentences from a Brown corpus section. They achieved 80.9%/80.3% recall/precision for section K, and 80.6%/81.3% for section N.[7] Our analogous method (TOP-Focus) achieved much better accuracy (3.7% and 4.9% better F_1, respectively).

In addition to portability experiments with the parsing model of (Collins, 1997), (Gildea, 2001) provided a comprehensive analysis of parser portability. On the basis of this analysis, a technique for parameter pruning was proposed leading to a significant reduction in the model size without a large decrease of accuracy. Gildea (2001) only reports results on sentences of 40 or less words on all the Brown corpus sections combined, for which he reports 80.3%/81.0% recall/precision when training only on data from the WSJ corpus, and 83.9%/84.8% when training on data from the WSJ corpus and all sections of the Brown corpus.

(Roark and Bacchiani, 2003) performed experiments on supervised and unsupervised PCFG adaptation to the target domain. They propose to use the statistics from a source domain to define priors over weights. However, in their experiments they used only trivial sub-cases of this approach, namely, count merging and model interpolation. They achieved very good improvement over their baseline and over (Gildea, 2001), but the absolute accuracies were still relatively low (as discussed above). They report results with combined Brown data (on sentences of 100 words or less), achieving 81.3%/80.9% when training only on the WSJ corpus and 85.4%/85.9% with their best method using the data from both domains.

7 Conclusions

This paper proposes a novel technique for improving parser portability, applying parse reranking with data-defined kernels. First a probabilistic model of parsing is trained on all the available data, including a large set of data from the source domain. This model is used to define a kernel over parse trees. Then this kernel is used in a large margin classifier trained on a small set of data only from the target domain. This classifier is used to rerank the top parses produced by the probabilistic model on the target domain. Experiments with a neural network statistical parser demonstrate that this approach leads to improved parser accuracy on the target domain, without any significant increase in computational cost.

[7] The sizes of Brown sections reported in (Ratnaparkhi, 1999) do not match the sizes of sections distributed in the Penn Treebank 3.0 package, so we couldn't replicate their split. We suspect that a preliminary version of the corpus was used for their experiments.

References

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures and the voted perceptron. In *Proc. ACL 2002* , pages 263–270, Philadelphia, PA.

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proc. ACL/EACL 1997* , pages 16–23, Somerset, New Jersey.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proc. EMNLP 2001* , Pittsburgh, PA.

James Henderson and Ivan Titov. 2005. Data-defi ned kernels for parse reranking derived from probabilistic models. In *Proc. ACL 2005* , Ann Arbor, MI.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proc. NAACL/HLT 2003* , pages 103–110, Edmonton, Canada.

Terry Koo and Michael Collins. 2005. Hidden-variable models for discriminative reranking. In *Proc. EMNLP 2005* , Vancouver, B.C., Canada.

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

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proc. ACL 2005* , Ann Arbor, MI.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–175.

Brian Roark and Michiel Bacchiani. 2003. Supervised and unsuperised PCFG adaptation to novel domains. In *Proc. HLT/ACL 2003* , Edmionton, Canada.

Libin Shen and Aravind K. Joshi. 2003. An SVM based voting algorithm with application to parse reranking. In *Proc. 7th Conf. on Computational Natural Language Learning*, pages 9–16, Edmonton, Canada.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. 21st Int. Conf. on Machine Learning*, pages 823–830, Banff, Alberta, Canada.

Alexander Yeh. 2000. More accurate tests for the statistical signifi cance of the result differences. In *Proc. 17th Int. Conf. on Computational Linguistics*, pages 947–953, Saarbruken, Germany.

Non-Local Modeling with a Mixture of PCFGs

Slav Petrov **Leon Barrett** **Dan Klein**
Computer Science Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720
{petrov, lbarrett, klein}@eecs.berkeley.edu

Abstract

While most work on parsing with PCFGs has focused on local correlations between tree configurations, we attempt to model non-local correlations using a finite mixture of PCFGs. A mixture grammar fit with the EM algorithm shows improvement over a single PCFG, both in parsing accuracy and in test data likelihood. We argue that this improvement comes from the learning of specialized grammars that capture non-local correlations.

1 Introduction

The probabilistic context-free grammar (PCFG) formalism is the basis of most modern statistical parsers. The symbols in a PCFG encode context-freedom assumptions about statistical dependencies in the derivations of sentences, and the relative conditional probabilities of the grammar rules induce scores on trees. Compared to a basic treebank grammar (Charniak, 1996), the grammars of high-accuracy parsers weaken independence assumptions by splitting grammar symbols and rules with either lexical (Charniak, 2000; Collins, 1999) or non-lexical (Klein and Manning, 2003; Matsuzaki et al., 2005) conditioning information. While such splitting, or conditioning, can cause problems for statistical estimation, it can dramatically improve the accuracy of a parser.

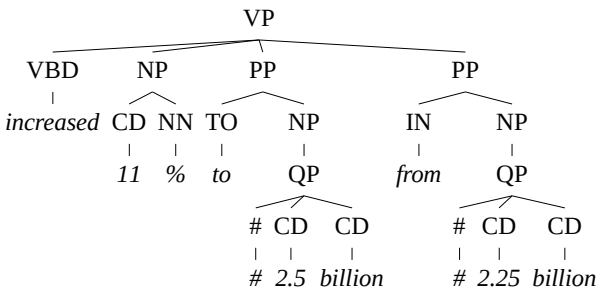
However, the configurations exploited in PCFG parsers are quite local: rules' probabilities may depend on parents or head words, but do not depend on arbitrarily distant tree configurations. For example, it is generally not modeled that if one quantifier phrase (QP in the Penn Treebank) appears in a sentence, the likelihood of finding another QP in that same sentence is greatly increased. This kind of effect is neither surprising nor unknown – for example, Bock and Loebell (1990) show experimentally that human language generation demonstrates priming effects. The mediating variables can not only include priming effects but also genre or stylistic conventions, as well as many other factors which are not adequately modeled by local phrase structure.

A reasonable way to add a latent variable to a generative model is to use a mixture of estimators, in this case a mixture of PCFGs (see Section 3). The general mixture of estimators approach was first suggested in the statistics literature by Titterington et al. (1962) and has since been adopted in machine learning (Ghahramani and Jordan, 1994). In a mixture approach, we have a new global variable on which all PCFG productions for a given sentence can be conditioned. In this paper, we experiment with a finite mixture of PCFGs. This is similar to the latent nonterminals used in Matsuzaki et al. (2005), but because the latent variable we use is global, our approach is more oriented toward learning non-local structure. We demonstrate that a mixture fit with the EM algorithm gives improved parsing accuracy and test data likelihood. We then investigate what is and is not being learned by the latent mixture variable. While mixture components are difficult to interpret, we demonstrate that the patterns learned are better than random splits.

2 Empirical Motivation

It is commonly accepted that the context freedom assumptions underlying the PCFG model are too

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 14–20, New York City, June 2006.



Rule	Score
QP → # CD CD	131.6
PRN → -LRB- ADJP -RRB	77.1
VP → VBD NP , PP PP	33.7
VP → VBD NP NP PP	28.4
PRN → -LRB- NP -RRB-	17.3
ADJP → QP	13.3
PP → IN NP ADVP	12.3
NP → NP PRN	12.3
VP → VBN PP PP PP	11.6
ADVP → NP RBR	10.1

Figure 1: Self-triggering: QP → # CD CD. If one British financial occurs in the sentence, the probability of seeing a second one in the same sentence is highly inreased. There is also a similar, but weaker, correlation for the American financial ($). On the right hand side we show the ten rules whose likelihoods are most increased in a sentence containing this rule.

strong and that weakening them results in better models of language (Johnson, 1998; Gildea, 2001; Klein and Manning, 2003). In particular, certain grammar productions often cooccur with other productions, which may be either near or distant in the parse tree. In general, there exist three types of correlations: (i) local (e.g. parent-child), (ii) non-local, and (iii) self correlations (which may be local or non-local).

In order to quantify the strength of a correlation, we use a likelihood ratio (LR). For two rules $\mathrm{X} \rightarrow \alpha$ and $\mathrm{Y} \rightarrow \beta$, we compute

$$LR(\mathrm{X} \rightarrow \alpha, \mathrm{Y} \rightarrow \beta) = \frac{\mathrm{P}(\alpha, \beta | X, Y)}{\mathrm{P}(\alpha | X, Y)\mathrm{P}(\beta | X, Y)}$$

This measures how much more often the rules occur together than they would in the case of independence. For rules that are correlated, this score will be high ($\gg 1$); if the rules are independent, it will be around 1, and if they are anti-correlated, it will be near 0.

Among the correlations present in the Penn Treebank, the local correlations are the strongest ones; they contribute 65% of the rule pairs with LR scores above 90 and 85% of those with scores over 200. Non-local and self correlations are in general common but weaker, with non-local correlations contributing approximately 85% of all correlations[1]. By adding a latent variable conditioning all productions, we aim to capture some of this interdependence between rules.

Correlations at short distances have been captured effectively in previous work (Johnson, 1998; Klein and Manning, 2003); vertical markovization (annotating nonterminals with their ancestor symbols) does this by simply producing a different distribution for each set of ancestors. This added context leads to substantial improvement in parsing accuracy. With local correlations already well captured, our main motivation for introducing a mixture of grammars is to capture long-range rule cooccurrences, something that to our knowledge has not been done successfully in the past.

As an example, the rule QP → # CD CD, representing a quantity of British currency, cooccurs with itself 132 times as often as if occurrences were independent. These cooccurrences appear in cases such as seen in Figure 1. Similarly, the rules VP → VBD NP PP , S and VP → VBG NP PP PP cooccur in the Penn Treebank 100 times as often as we would expect if they were independent. They appear in sentences of a very particular form, telling of an action and then giving detail about it; an example can be seen in Figure 2.

3 Mixtures of PCFGs

In a probabilistic context-free grammar (PCFG), each rule $\mathrm{X} \rightarrow \alpha$ is associated with a conditional probability $\mathrm{P}(\alpha | X)$ (Manning and Schütze, 1999). Together, these rules induce a distribution over trees $\mathrm{P}(T)$. A mixture of PCFGs enriches the basic model

[1]Quantifying the amount of non-local correlation is problematic; most pairs of cooccuring rules are non-local and will, due to small sample effects, have LR ratios greater than 1 even if they were truly independent in the limit.

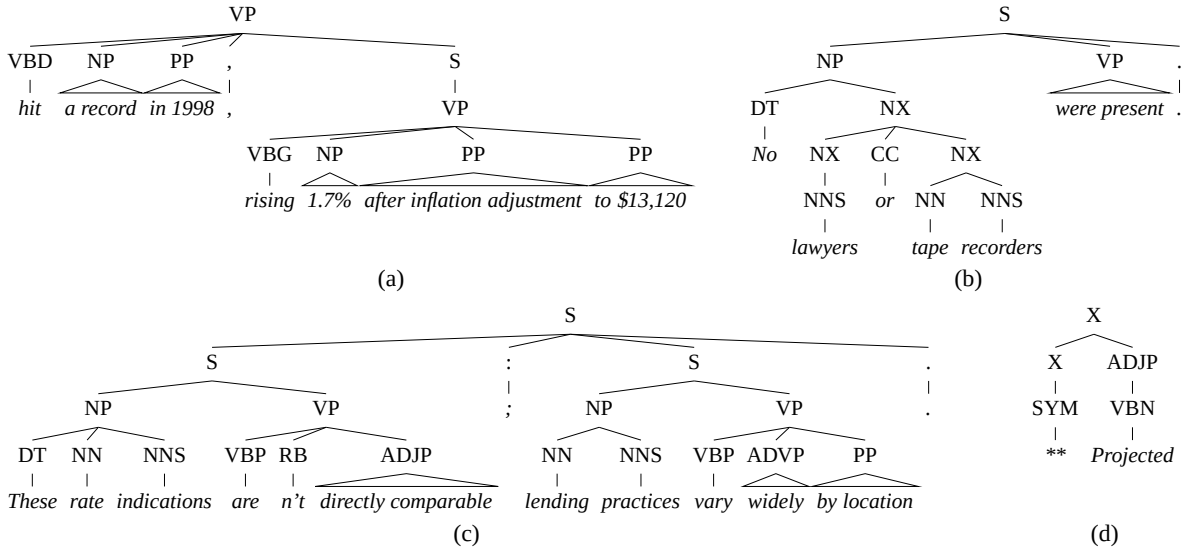


Figure 2: Tree fragments demonstrating coocurrences. (a) and (c) Repeated formulaic structure in one grammar: rules VP → VBD NP PP , S and VP → VBG NP PP PP and rules VP → VBP RB ADJP and VP → VBP ADVP PP. (b) Sibling effects, though not parallel structure, rules: NX → NNS and NX → NN NNS. (d) A special structure for footnotes has rules ROOT → X and X → SYM coocurring with high probability.

by allowing for multiple grammars, G_i, which we call *individual grammars*, as opposed to a single grammar. Without loss of generality, we can assume that the individual grammars share the same set of rules. Therefore, each original rule $\mathrm{X} \rightarrow \alpha$ is now associated with a vector of probabilities, $\mathrm{P}(\alpha|X, i)$. If, in addition, the individual grammars are assigned prior probabilities $\mathrm{P}(i)$, then the entire mixture induces a joint distribution over *derivations* $\mathrm{P}(T, i) = \mathrm{P}(i)\mathrm{P}(T|i)$ from which we recover a distribution over trees by summing over the grammar index i.

As a generative derivation process, we can think of this in two ways. First, we can imagine G to be a latent variable on which all productions are conditioned. This view emphasizes that any otherwise unmodeled variable or variables can be captured by the latent variable G. Second, we can imagine selecting an individual grammar G_i and then generating a sentence using that grammar. This view is associated with the expectation that there are multiple grammars for a language, perhaps representing different genres or styles. Formally, of course, the two views are the same.

3.1 Hierarchical Estimation

So far, there is nothing in the formal mixture model to say that rule probabilities in one component have any relation to those in other components. However, we have a strong intuition that many rules, such as NP → DT NN, will be common in all mixture components. Moreover, we would like to pool our data across components when appropriate to obtain more reliable estimators.

This can be accomplished with a hierarchical estimator for the rule probabilities. We introduce a *shared grammar* G_s. Associated to each rewrite is now a latent variable $L = \{\mathrm{S}, \mathrm{I}\}$ which indicates whether the used rule was derived from the shared grammar G_s or one of the individual grammars G_i:

$$\mathrm{P}(\alpha|X, i) = \lambda \mathrm{P}(\alpha|X, i, \ell{=}\mathrm{I}) + (1 - \lambda)\mathrm{P}(\alpha|X, i, \ell{=}\mathrm{S}),$$

where $\lambda \equiv P(\ell = \mathrm{I})$ is the probability of choosing the individual grammar and can also be viewed as a mixing coefficient. Note that $\mathrm{P}(\alpha|X, i, \ell{=}\mathrm{S}) = \mathrm{P}(\alpha|X, \ell{=}\mathrm{S})$, since the shared grammar is the same for all individual grammars. This kind of hierarchical estimation is analogous to that used in hierarchical mixtures of naive-Bayes for

text categorization (McCallum et al., 1998).

The hierarchical estimator is most easily described as a generative model. First, we choose a individual grammar G_i. Then, for each nonterminal, we select a level from the back-off hierarchy grammar: the individual grammar G_i with probability λ, and the shared grammar G_s with probability $1 - \lambda$. Finally, we select a rewrite from the chosen level. To emphasize: the derivation of a phrase-structure tree in a hierarchically-estimated mixture of PCFGs involves two kinds of hidden variables: the grammar G used for each sentence, and the level L used at each tree node. These hidden variables will impact both learning and inference in this model.

3.2 Inference: Parsing

Parsing involves inference for a given sentence S. One would generally like to calculate the *most probable parse* – that is, the tree T which has the highest probability $\mathrm{P}(T|S) \propto \sum_i \mathrm{P}(i)\mathrm{P}(T|i)$. However, this is difficult for mixture models. For a single grammar we have:

$$\mathrm{P}(T, i) = \mathrm{P}(i) \prod_{X \to \alpha \in T} \mathrm{P}(\alpha|X, i).$$

This score decomposes into a product and it is simple to construct a dynamic programming algorithm to find the optimal T (Baker, 1979). However, for a mixture of grammars we need to sum over the individual grammars:

$$\sum_i \mathrm{P}(T, i) = \sum_i \mathrm{P}(i) \prod_{X \to \alpha \in T} \mathrm{P}(\alpha|X, i).$$

Because of the outer sum, this expression unfortunately does not decompose into a product over scores of subparts. In particular, a tree which maximizes the sum need not be a top tree for any single component.

As is true for many other grammar formalisms in which there is a derivation / parse distinction, an alternative to finding the most probable parse is to find the *most probable derivation* (Vijay-Shankar and Joshi, 1985; Bod, 1992; Steedman, 2000). Instead of finding the tree T which maximizes $\sum_i \mathrm{P}(T, i)$, we find both the tree T and component i which maximize $\mathrm{P}(T, i)$. The most probable derivation can be found by simply doing standard PCFG parsing once for each component, then comparing the resulting trees' likelihoods.

3.3 Learning: Training

Training a mixture of PCFGs from a treebank is an incomplete data problem. We need to decide which individual grammar gave rise to a given observed tree. Moreover, we need to select a generation path (individual grammar or shared grammar) for each rule in the tree. To learn estimate parameters, we can use a standard Expectation-Maximization (EM) approach.

In the E-step, we compute the posterior distributions of the latent variables, which are in this case both the component G of each sentence and the hierarchy level L of each rewrite. Note that, unlike during parsing, there is no uncertainty over the actual rules used, so the E-step does not require summing over possible trees. Specifically, for the variable G we have

$$\mathrm{P}(i|T) = \frac{\mathrm{P}(T, i)}{\sum_j \mathrm{P}(T, j)}.$$

For the hierarchy level L we can write

$$\mathrm{P}(\ell = \textsc{i}|X \to \alpha, i, T) = \frac{\lambda \mathrm{P}(\alpha|X, \ell{=}\textsc{i})}{\lambda \mathrm{P}(\alpha|X, i, \ell{=}\textsc{i}) + (1 - \lambda)\mathrm{P}(\alpha|X, \ell{=}\textsc{s})},$$

where we slightly abuse notation since the rule $X \to \alpha$ can occur multiple times in a tree T.

In the M-step, we find the maximum-likelihood model parameters given these posterior assignments; i.e., we find the best grammars given the way the training data's rules are distributed between individual and shared grammars. This is done exactly as in the standard single-grammar model using relative expected frequencies. The updates are shown in Figure 3.3, where $\mathbf{T} = \{T_1, T_2, \dots\}$ is the training set.

We initialize the algorithm by setting the assignments from sentences to grammars to be uniform between all the individual grammars, with a small random perturbation to break symmetry.

4 Results

We ran our experiments on the Wall Street Journal (WSJ) portion of the Penn Treebank using the standard setup: We trained on sections 2 to 21, and we used section 22 as a validation set for tuning model hyperparameters. Results are reported

$$\mathrm{P}(i) \leftarrow \frac{\sum_{T_k \in \mathbf{T}} \mathrm{P}(i|T_k)}{\sum_i \sum_{T_k \in \mathbf{T}} \mathrm{P}(i|T_k)} = \frac{\sum_{T_k \in \mathbf{T}} \mathrm{P}(i|T_k)}{k}$$

$$\mathrm{P}(l = \textsc{i}) \leftarrow \frac{\sum_{T_k \in \mathbf{T}} \sum_{X \to \alpha \in T_k} \mathrm{P}(\ell = \textsc{i} | X \to \alpha)}{\sum_{T_k \in \mathbf{T}} |T_k|}$$

$$\mathrm{P}(\alpha | X, i, \ell = \textsc{i}) \leftarrow \frac{\sum_{T_k \in \mathbf{T}} \sum_{X \to \alpha \in T_k} \mathrm{P}(i|T_k) \mathrm{P}(\ell = \textsc{i} | T_k, i, X \to \alpha)}{\sum_{\alpha'} \sum_{T_k \in \mathbf{T}} \sum_{X \to \alpha' \in T_k} \mathrm{P}(i|T_k) \mathrm{P}(\ell = \textsc{i} | T_k, i, X \to \alpha')}$$

Figure 3: Parameter updates. The shared grammar's parameters are re-estimated in the same manner.

on all sentences of 40 words or less from section 23. We use a markovized grammar which was annotated with parent and sibling information as a baseline (see Section 4.2). Unsmoothed maximum-likelihood estimates were used for rule probabilities as in Charniak (1996). For the tagging probabilities, we used maximum-likelihood estimates for $\mathrm{P}(tag|word)$. Add-one smoothing was applied to unknown and rare (seen ten times or less during training) words before inverting those estimates to give $\mathrm{P}(word|tag)$. Parsing was done with a simple Java implementation of an agenda-based chart parser.

4.1 Parsing Accuracy

The EM algorithm is guaranteed to continuously increase the likelihood on the training set until convergence to a local maximum. However, the likelihood on unseen data will start decreasing after a number of iterations, due to overfitting. This is demonstrated in Figure 4. We use the likelihood on the validation set to stop training before overfitting occurs.

In order to evaluate the performance of our model, we trained mixture grammars with various numbers of components. For each configuration, we used EM to obtain twelve estimates, each time with a different random initialization. We show the F_1-score for the model with highest log-likelihood on the validation set in Figure 4. The results show that a mixture of grammars outperforms a standard, single grammar PCFG parser.[2]

4.2 Capturing Rule Correlations

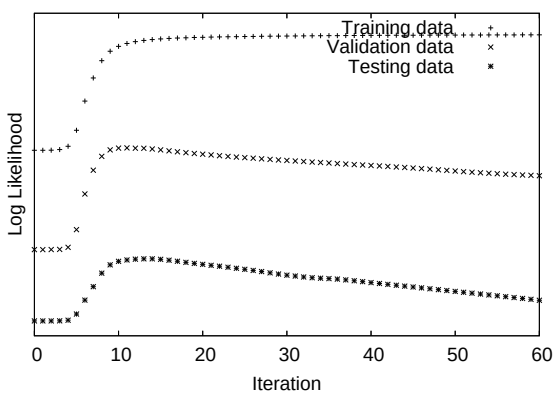
As described in Section 2, we hope that the mixture model will capture long-range correlations in the data. Since local correlations can be captured by adding parent annotation, we combine our mixture model with a grammar in which node probabilities depend on the parent (the last vertical ancestor) and the closest sibling (the last horizontal ancestor). Klein and Manning (2003) refer to this grammar as a markovized grammar of vertical order = 2 and horizontal order = 1. Because many local correlations are captured by the markovized grammar, there is a greater hope that observed improvements stem from non-local correlations.

In fact, we find that the mixture does capture non-local correlations. We measure the degree to which a grammar captures correlations by calculating the total squared error between LR scores of the grammar and corpus, weighted by the probability of seeing nonterminals. This is 39422 for a single PCFG, but drops to 37125 for a mixture with five individual grammars, indicating that the mixture model better captures the correlations present in the corpus. As a concrete example, in the Penn Treebank, we often see the rules FRAG $\rightarrow$ ADJP and PRN $\rightarrow$, SBAR , cooccurring; their LR is 134. When we learn a single markovized PCFG from the treebank, that grammar gives a likelihood ratio of only 61. However, when we train with a hierarchical model composed of a shared grammar and four individual grammars, we find that the grammar likelihood ratio for these rules goes up to 126, which is very similar to that of the empirical ratio.

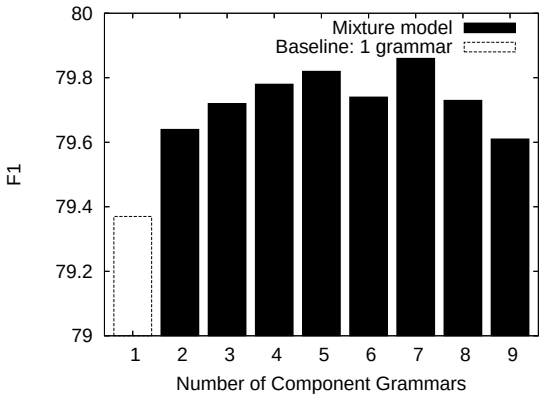
4.3 Genre

The mixture of grammars model can equivalently be viewed as capturing either non-local correlations or variations in grammar. The latter view suggests that the model might benefit when the syntactic structure

[2] This effect is statistically significant.



(a)



(b)

Figure 4: (a) Log likelihood of training, validation, and test data during training (transformed to fit on the same plot). Note that when overfitting occurs the likelihood on the validation and test data starts decreasing (after 13 iterations). (b) The accuracy of the mixture of grammars model with $\lambda = 0.4$ versus the number of grammars. Note the improvement over a 1-grammar PCFG model.

varies significantly, as between different genres. We tested this with the Brown corpus, of which we used 8 different genres (*f*, *g*, *k*, *l*, *m*, *n*, *p*, and *r*). We follow Gildea (2001) in using the ninth and tenth sentences of every block of ten as validation and test data, respectively, because a contiguous test section might not be representative due to the genre variation.

To test the effects of genre variation, we evaluated various training schemes on the Brown corpus. The single grammar baseline for this corpus gives $F_1 = 79.75$, with log likelihood (LL) on the testing data=-242561. The first test, then, was to estimate each individual grammar from only one genre. We did this by assigning sentences to individual grammars by genre, without using any EM training. This increases the data likelihood, though it reduces the F_1 score ($F_1 = 79.48$, LL=-242332). The increase in likelihood indicates that there *are* genre-specific features that our model can represent. (The lack of F_1 improvement may be attributed to the increased difficulty of estimating rule probabilities after dividing the already scant data available in the Brown corpus. This small quantity of data makes overfitting almost certain.)

However, local minima and lack of data cause difficulty in learning genre-specific features. If we start with sentences assigned by genre as before, but then train with EM, both F_1 and test data log likelihood drop ($F_1 = 79.37$, LL=-242100). When we use EM with a random initialization, so that sentences are not assigned directly to grammars, the scores go down even further ($F_1 = 79.16$, LL=-242459). This indicates that the model can capture variation between genres, but that maximum training data likelihood does not necessarily give maximum accuracy. Presumably, with more genre-specific data available, learning would generalize better. So, genre-specific grammar variation is real, but it is difficult to capture via EM.

4.4 Smoothing Effects

While the mixture of grammars captures rule correlations, it may also enhance performance via smoothing effects. Splitting the data randomly could produce a smoothed shared grammar, G_s, that is a kind of held-out estimate which could be superior to the unsmoothed ML estimates for the single-component grammar.

We tested the degree of generalization by evaluating the shared grammar alone and also a mixture of the shared grammar with the known single grammar. Those shared grammars were extracted after training the mixture model with four individual grammars. We found that both the shared grammar alone (F_1=79.13, LL=-333278) and the shared grammar mixed with the single grammar (F_1=79.36, LL=-331546) perform worse than a sin-

gle PCFG (F_1=79.37, LL=-327658). This indicates that smoothing is not the primary learning effect contributing to increased F_1.

5 Conclusions

We examined the sorts of rule correlations that may be found in natural language corpora, discovering non-local correlations not captured by traditional models. We found that using a model capable of representing these non-local features gives improvement in parsing accuracy and data likelihood. This improvement is modest, however, primarily because local correlations are so much stronger than non-local ones.

References

J. Baker. 1979. Trainable grammars for speech recognition. *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550.

K. Bock and H. Loebell. 1990. Framing sentences. *Cognition*, 35:1–39.

R. Bod. 1992. A computational model of language performance: Data oriented parsing. *International Conference on Computational Linguistics (COLING)*.

E. Charniak. 1996. Tree-bank grammars. In *Proc. of the 13th National Conference on Artificial Intelligence (AAAI)*, pages 1031–1036.

E. Charniak. 2000. A maximum–entropy–inspired parser. In *Proc. of the Conference of the North American chapter of the Association for Computational Linguistics (NAACL)*, pages 132–139.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Univ. of Pennsylvania.

Z. Ghahramani and M. I. Jordan. 1994. Supervised learning from incomplete data via an EM approach. In *Advances in Neural Information Processing Systems (NIPS)*, pages 120–127.

D. Gildea. 2001. Corpus variation and parser performance. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

M. Johnson. 1998. Pcfg models of linguistic tree representations. *Computational Linguistics*, 24:613–632.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. *Proc. of the 41st Meeting of the Association for Computational Linguistics (ACL)*, pages 423–430.

C. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proc. of the 43rd Meeting of the Association for Computational Linguistics (ACL)*, pages 75–82.

A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng. 1998. Improving text classification by shrinkage in a hierarchy of classes. In *Int. Conf. on Machine Learning (ICML)*, pages 359–367.

M. Steedman. 2000. *The Syntactic Process.* The MIT Press, Cambridge, Massachusetts.

D. Titterington, A. Smith, and U. Makov. 1962. *Statistical Analysis of Finite Mixture Distributions*. Wiley.

K. Vijay-Shankar and A. Joshi. 1985. Some computational properties of tree adjoining grammars. *Proc. of the 23th Meeting of the Association for Computational Linguistics (ACL)*, pages 82–93.

Improved Large Margin Dependency Parsing via Local Constraints and Laplacian Regularization

Qin Iris Wang **Colin Cherry** **Dan Lizotte** **Dale Schuurmans**
Department of Computing Science
University of Alberta
{wqin,colinc,dlizotte,dale}@cs.ualberta.ca

Abstract

We present an improved approach for learning dependency parsers from treebank data. Our technique is based on two ideas for improving large margin training in the context of dependency parsing. First, we incorporate local constraints that enforce the correctness of each individual link, rather than just scoring the global parse tree. Second, to cope with sparse data, we smooth the lexical parameters according to their underlying word similarities using Laplacian Regularization. To demonstrate the benefits of our approach, we consider the problem of parsing Chinese treebank data using only lexical features, that is, without part-of-speech tags or grammatical categories. We achieve state of the art performance, improving upon current large margin approaches.

1 Introduction

Over the past decade, there has been tremendous progress on learning parsing models from treebank data (Collins, 1997; Charniak, 2000; Wang et al., 2005; McDonald et al., 2005). Most of the early work in this area was based on postulating *generative* probability models of language that included parse structure (Collins, 1997). Learning in this context consisted of estimating the parameters of the model with simple likelihood based techniques, but incorporating various smoothing and back-off estimation tricks to cope with the sparse data problems (Collins, 1997; Bikel, 2004). Subsequent research began to focus more on *conditional* models of parse structure given the input sentence, which allowed discriminative training techniques such as maximum conditional likelihood (i.e. "maximum entropy") to be applied (Ratnaparkhi, 1999; Charniak, 2000). In fact, recently, effective conditional parsing models have been learned using relatively straightforward "plug-in" estimates, augmented with similarity based smoothing (Wang et al., 2005). Currently, the work on conditional parsing models appears to have culminated in large margin training (Taskar et al., 2003; Taskar et al., 2004; Tsochantaridis et al., 2004; McDonald et al., 2005), which currently demonstrates the state of the art performance in English dependency parsing (McDonald et al., 2005).

Despite the realization that maximum margin training is closely related to maximum conditional likelihood for conditional models (McDonald et al., 2005), a sufficiently unified view has not yet been achieved that permits the easy exchange of improvements between the probabilistic and non-probabilistic approaches. For example, smoothing methods have played a central role in probabilistic approaches (Collins, 1997; Wang et al., 2005), and yet they are not being used in current large margin training algorithms. However, as we demonstrate, not only can smoothing be applied in a large margin training framework, it leads to generalization improvements in much the same way as probabilistic approaches. The second key observation we make is somewhat more subtle. It turns out that probabilistic approaches pay closer attention to the individual errors made by each component of a parse, whereas the training error minimized in the large margin approach—the "structured margin loss" (Taskar et al., 2003; Tsochantaridis et al., 2004; McDonald et al., 2005)—is a coarse measure that only assesses the total error of an entire parse rather than focusing on the error of any particular component.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 21–28, New York City, June 2006.

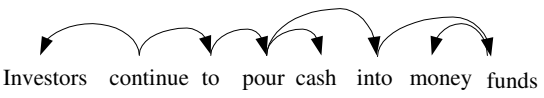


Figure 1: A dependency tree

In this paper, we make two contributions to the large margin approach to learning parsers from supervised data. First, we show that smoothing based on lexical similarity is not only possible in the large margin framework, but more importantly, allows better generalization to new words not encountered during training. Second, we show that the large margin training objective can be significantly refined to assess the error of each component of a given parse, rather than just assess a global score. We show that these two extensions together lead to greater training accuracy and better generalization to novel input sentences than current large margin methods.

To demonstrate the benefit of combining useful learning principles from both the probabilistic and large margin frameworks, we consider the problem of learning a dependency parser for Chinese. This is an interesting test domain because Chinese does not have clearly defined parts-of-speech, which makes lexical smoothing one of the most natural approaches to achieving reasonable results (Wang et al., 2005).

2 Lexicalized Dependency Parsing

A dependency tree specifies which words in a sentence are directly related. That is, the dependency structure of a sentence is a directed tree where the nodes are the words in the sentence and links represent the direct dependency relationships between the words; see Figure 1. There has been a growing interest in dependency parsing in recent years. (Fox, 2002) found that the dependency structures of a pair of translated sentences have a greater degree of cohesion than phrase structures. (Cherry and Lin, 2003) exploited such cohesion between the dependency structures to improve the quality of word alignment of parallel sentences. Dependency relations have also been found to be useful in information extraction (Culotta and Sorensen, 2004; Yangarber et al., 2000).

A key aspect of a dependency tree is that it does not necessarily report parts-of-speech or phrase labels. Not requiring parts-of-speech is especially beneficial for languages such as Chinese, where parts-of-speech are not as clearly defined as English. In Chinese, clear indicators of a word's part-of-speech such as suffixes "-ment", "-ous" or function words such as "the", are largely absent. One of our motivating goals is to develop an approach to learning dependency parsers that is strictly lexical. Hence the parser can be trained with a treebank that only contains the dependency relationships, making annotation much easier.

Of course, training a parser with bare word-to-word relationships presents a serious challenge due to data sparseness. It was found in (Bikel, 2004) that Collins' parser made use of bi-lexical statistics only 1.49% of the time. The parser has to compute back-off probability using parts-of-speech in vast majority of the cases. In fact, it was found in (Gildea, 2001) that the removal of bi-lexical statistics from a state of the art PCFG parser resulted in very little change in the output. (Klein and Manning, 2003) presented an unlexicalized parser that eliminated all lexicalized parameters. Its performance was close to the state of the art lexicalized parsers.

Nevertheless, in this paper we follow the recent work of (Wang et al., 2005) and consider a completely lexicalized parser that uses no parts-of-speech or grammatical categories of any kind. Even though a part-of-speech lexicon has always been considered to be necessary in any natural language parser, (Wang et al., 2005) showed that distributional word similarities from a large unannotated corpus can be used to supplant part-of-speech smoothing with word similarity smoothing, to still achieve state of the art dependency parsing accuracy for Chinese.

Before discussing our modifications to large margin training for parsing in detail, we first present the dependency parsing model we use. We then give a brief overview of large margin training, and then present our two modifications. Subsequently, we present our experimental results on fully lexical dependency parsing for Chinese.

3 Dependency Parsing Model

Given a sentence $W = (w_1, ..., w_n)$ we are interested in computing a directed dependency tree,

T, over W. In particular, we assume that a directed dependency tree T consists of ordered pairs $(w_i \rightarrow w_j)$ of words in W such that each word appears in at least one pair and each word has in-degree at most one. Dependency trees are usually assumed to be projective (no crossing arcs), which means that if there is an arc $(w_i \rightarrow w_j)$, then w_i is an ancestor of all the words between w_i and w_j. Let $\Phi(W)$ denote the set of all the directed, projective trees that span W.

Given an input sentence W, we would like to be able to compute the best parse; that is, a projective tree, $T \in \Phi(W)$, that obtains the highest "score". In particular, we follow (Eisner, 1996; Eisner and Satta, 1999; McDonald et al., 2005) and assume that the score of a complete spanning tree T for a given sentence, whether probabilistically motivated or not, can be decomposed as a sum of local scores for each link (a word pair). In which case, the parsing problem reduces to

$$T^* = \arg \max_{T \in \Phi(W)} \sum_{(w_i \rightarrow w_j) \in T} s(w_i \rightarrow w_j) \quad (1)$$

where the score $s(w_i \rightarrow w_j)$ can depend on any measurable property of w_i and w_j within the tree T. This formulation is sufficiently general to capture most dependency parsing models, including probabilistic dependency models (Wang et al., 2005; Eisner, 1996) as well as non-probabilistic models (McDonald et al., 2005). For standard scoring functions, parsing requires an $O(n^3)$ dynamic programming algorithm to compute a projective tree that obtains the maximum score (Eisner and Satta, 1999; Wang et al., 2005; McDonald et al., 2005).

For the purpose of learning, we decompose each link score into a weighted linear combination of features

$$s(w_i \rightarrow w_j) = \boldsymbol{\theta}^\top \mathbf{f}(w_i \rightarrow w_j) \quad (2)$$

where $\boldsymbol{\theta}$ are the weight parameters to be estimated during training.

Of course, the specific features used in any real situation are critical for obtaining a reasonable dependency parser. The natural sets of features to consider in this setting are very large, consisting at the very least of features indexed by all possible lexical items (words). For example, natural features to use for dependency parsing are indicators of each possible word pair

$$f_{uv}(w_i \rightarrow w_j) = 1_{(w_i = u)} 1_{(w_j = v)}$$

which allows one to represent the tendency of two words, u and v, to be directly linked in a parse. In this case, there is a corresponding parameter θ_{uv} to be learned for each word pair, which represents the strength of the possible linkage.

A large number of features leads to a serious risk of over-fitting due to sparse data problems. The standard mechanisms for mitigating such effects are to combine features via *abstraction* (e.g. using parts-of-speech) or *smoothing* (e.g. using word similarity based smoothing). For abstraction, a common strategy is to use parts-of-speech to compress the feature set, for example by only considering the tag of the parent

$$f_{pv}(w_i \rightarrow w_j) = 1_{(pos(w_i) = p)} 1_{(w_j = v)}$$

However, rather than use abstraction, we will follow a purely lexical approach and only consider features that are directly computable from the words themselves (or statistical quantities that are directly measurable from these words).

In general, the most important aspect of a link feature is simply that it measures something about a candidate word pair that is predictive of whether the words will actually be linked in a given sentence. Thus, many other natural features, beyond parts-of-speech and abstract grammatical categories, immediately suggest themselves as being predictive of link existence. For example, one very useful feature is simply the degree of association between the two words as measured by their pointwise mutual information

$$f_{PMI}(w_i \rightarrow w_j) = PMI(w_i, w_j)$$

(We describe in Section 6 below how we compute this association measure on an auxiliary corpus of unannotated text.) Another useful link feature is simply the distance between the two words in the sentence; that is, how many words they have between them

$$f_{dist}(w_i \rightarrow w_j) = |position(w_i) - position(w_j)|$$

In fact, the likelihood of a direct link between two words diminishes quickly with distance, which motivates using more rapidly increasing functions of distance, such as the square

$$f_{dist2}(w_i \rightarrow w_j) = (position(w_i) - position(w_j))^2$$

In our experiments below, we used only these simple, lexically determined features, $\{f_{uv}\}$, f_{PMI}, f_{dist} and f_{dist2}, *without* the parts-of-speech $\{f_{pv}\}$. Currently, we only use undirected forms of these features, where, for example, $f_{uv} = f_{vu}$ for all pairs (or, put another way, we tie the parameters $\theta_{uv} = \theta_{vu}$ together for all u, v). Ideally, we would like to use directed features, but we have already found that these simple undirected features permit state of the art accuracy in predicting (undirected) dependencies. Nevertheless, extending our approach to directed features and contextual features, as in (Wang et al., 2005), remains an important direction for future research.

4 Large Margin Training

Given a training set of sentences annotated with their correct dependency parses, $(W_1, T_1), ..., (W_N, T_N)$, the goal of learning is to estimate the parameters of the parsing model, $\boldsymbol{\theta}$. In particular, we seek values for the parameters that can accurately reconstruct the training parses, but more importantly, are also able to accurately predict the dependency parse structure on future test sentences.

To train $\boldsymbol{\theta}$ we follow the large margin training approach of (Taskar et al., 2003; Tsochantaridis et al., 2004), which has been applied with great success to dependency parsing (Taskar et al., 2004; McDonald et al., 2005). Large margin training can be expressed as minimizing a regularized loss (Hastie et al., 2004)

$$\min_{\boldsymbol{\theta}} \frac{\beta}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta} \quad + \quad \sum_i \max_{L_i} \Delta(L_i, T_i) - (s(\boldsymbol{\theta}, T_i) - s(\boldsymbol{\theta}, L_i)) \tag{3}$$

where T_i is the target tree for sentence W_i; L_i ranges over all possible alternative trees in $\Phi(W_i)$; $s(\boldsymbol{\theta}, T) = \sum_{(w_i \rightarrow w_j) \in T} \boldsymbol{\theta}^\top \mathbf{f}(w_i \rightarrow w_j)$; and $\Delta(L_i, T_i)$ is a measure of distance between the two trees L_i and T_i.

Using the techniques of (Hastie et al., 2004) one can show that minimizing (4) is equivalent to solving the quadratic program

$$\min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \quad \frac{\beta}{2}\boldsymbol{\theta}^\top\boldsymbol{\theta} + \mathbf{e}^\top\boldsymbol{\xi} \quad \text{subject to} \tag{4}$$
$$\xi_i \geq \Delta(T_i, L_i) + s(\boldsymbol{\theta}, L_i) - s(\boldsymbol{\theta}, T_i)$$
$$\text{for all } i, L_i \in \Phi(W_i)$$

which corresponds to the training problem posed in (McDonald et al., 2005).

Unfortunately, the quadratic program (4) has three problems one must address. First, there are exponentially many constraints—corresponding to each possible parse of each training sentence—which forces one to use alternative training procedures, such as incremental constraint generation, to slowly converge to a solution (McDonald et al., 2005; Tsochantaridis et al., 2004). Second, and related, the original loss (4) is only evaluated at the global parse tree level, and is not targeted at penalizing any specific component in an incorrect parse. Although (McDonald et al., 2005) explicitly describes this as an advantage over previous approaches (Ratnaparkhi, 1999; Yamada and Matsumoto, 2003), below we find that changing the loss to enforce a more detailed set of constraints leads to a more effective approach. Third, given the large number of bi-lexical features $\{f_{uv}\}$ in our model, solving (4) directly will over-fit any reasonable training corpus. (Moreover, using a large β to shrink the $\boldsymbol{\theta}$ values does not mitigate the sparse data problem introduced by having so many features.) We now present our refinements that address each of these issues in turn.

5 Training with Local Constraints

We are initially focusing on training on just an undirected link model, where each parameter in the model is a weight $\theta_{ww'}$ between two words, w and w', respectively. Since links are undirected, these weights are symmetric $\theta_{ww'} = \theta_{w'w}$, and we can also write the score in an undirected fashion as: $s(w, w') = \boldsymbol{\theta}^\top \mathbf{f}(w, w')$. The main advantage of working with the undirected link model is that the constraints needed to ensure correct parses on the training data are *much* easier to specify in this case. Ignoring the projective (no crossing arcs) constraint for the moment, an undirected dependency parse can

be equated with a maximum score spanning tree of a sentence. Given a target parse, the set of constraints needed to ensure the target parse is in fact the maximum score spanning tree under the weights $\boldsymbol{\theta}$, by at least a minimum amount, is a simple set of linear constraints: for any edge $w_1 w_2$ that is *not* in the target parse, one simply adds two constraints

$$\begin{aligned} \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_1') &\geq \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_2) + 1 \\ \boldsymbol{\theta}^\top \mathbf{f}(w_2, w_2') &\geq \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_2) + 1 \end{aligned} \quad (5)$$

where the edges $w_1 w_1'$ and $w_2 w_2'$ are the adjacent edges that actually occur in the target parse that are also on the path between w_1 and w_2. (These would have to be the only such edges, or there would be a loop in the parse tree.) These constraints behave very naturally by forcing the weight of an omitted edge to be smaller than the adjacent included edges that would form a loop, which ensures that the omitted edge would not be added to the maximum score spanning tree before the included edges.

In this way, one can simply accumulate the set of linear constraints (5) for every edge that fails to be included in the target parse for the sentences where it is a candidate. We denote this set of constraints by

$$A = \{\boldsymbol{\theta}^\top \mathbf{f}(w_1, w_1') \geq \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_2) + 1\}$$

Importantly, the constraint set A is *convex* in the link weight parameters $\boldsymbol{\theta}$, as it consists only of linear constraints.

Ignoring the non-crossing condition, the constraint set A is exact. However, because of the non-crossing condition, the constraint set A is more restrictive than necessary. For example, consider the word sequence $...w_i w_{i+1} w_{i+2} w_{i+3}...$, where the edge $w_{i+1} w_{i+3}$ is in the target parse. Then the edge $w_i w_{i+2}$ can be ruled out of the parse in one of two ways: it can be ruled out by making its score less than the adjacent scores as specified in (5), *or* it can be ruled out by making its score smaller than the score of $w_{i+1} w_{i+3}$. Thus, the exact constraint contains a disjunction of two different constraints, which creates a *non*-convex constraint in $\boldsymbol{\theta}$. (The union of two convex sets is not necessarily convex.) This is a weakening of the original constraint set A. Unfortunately, this means that, given a large training corpus, the constraint set A can easily become infeasible.

Nevertheless, the constraints in A capture much of the relevant structure in the data, and are easy to enforce. Therefore, we wish to maintain them. However, rather than impose the constraints exactly, we enforce them approximately through the introduction of slack variables $\boldsymbol{\xi}$. The relaxed constraints can then be expressed as

$$\boldsymbol{\theta}^\top \mathbf{f}(w_1, w_1') \geq \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_2) + 1 - \xi_{w_1 w_2, w_1 w_1'} \quad (6)$$

and therefore a maximum soft margin solution can then be expressed as a quadratic program

$$\begin{aligned} \min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \quad & \frac{\beta}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta} + \boldsymbol{\xi}^\top \mathbf{e} \quad \text{subject to} \\ & \{\boldsymbol{\theta}^\top \mathbf{f}(w_1, w_1') \geq \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_2) + 1 - \xi_{w_1 w_2, w_1 w_1'}\} \\ & \text{for all constraints in } A \end{aligned} \quad (7)$$

where $\mathbf{e}$ denotes the vector of all 1's.

Even though the slacks are required because we have slightly over-constrained the parameters, given that there are so many parameters and a sparse data problem as well, it seems desirable to impose a stronger set of constraints. A set of solution parameters achieved in this way will allow maximum weight spanning trees to correctly parse nearly all of the training sentences, even without the non-crossing condition (see the results in Section 8).

This quadratic program has the advantage of producing link parameters that will correctly parse most of the training data. Unfortunately, the main drawback of this method thus far is that it does not offer any mechanism by which the link weights $\theta_{ww'}$ can be *generalized* to new or rare words. Given the sparse data problem, some form of generalization is necessary to achieve good test results. We achieve this by exploiting distributional similarities between words to smooth the parameters.

6 Distributional Word Similarity

Treebanks are an extremely precious resource. The average cost of producing a treebank parse can run as high as 30 person-minutes per sentence (20 words on average). Similarity-based smoothing, on the other hand, allows one to tap into auxiliary sources of raw unannotated text, which is practically unlimited. With this extra data, one can estimate parameters for words that have never appeared in the training corpus.

The basic intuition behind similarity smoothing is that words that tend to appear in the same contexts tend to have similar meanings. This is known as the Distributional Hypothesis in linguistics (Harris, 1968). For example, the words test and exam are similar because both of them can follow verbs such as administer, cancel, cheat on, conduct, etc.

Many methods have been proposed to compute distributional similarity between words, e.g., (Hindle, 1990; Pereira et al., 1993; Grefenstette, 1994; Lin, 1998). Almost all of the methods represent a word by a feature vector where each feature corresponds to a type of context in which the word appeared. They differ in how the feature vectors are constructed and how the similarity between two feature vectors is computed.

In our approach below, we define the features of a word w to be the set of words that occurred within a small window of w in a large corpus. The context window of w consists of the closest non-stop-word on each side of w and the stop-words in between. The value of a feature w' is defined as the pointwise mutual information between the w' and w: $\text{PMI}(w', w) = \log(\frac{P(w,w')}{P(w)P(w')})$. The similarity between two words, $S(w_1, w_2)$, is then defined as the cosine of the angle between their feature vectors.

We use this similarity information both in training and in parsing. For training, we smooth the parameters according to their underlying word-*pair* similarities by introducing a Laplacian regularizer, which will be introduced in the next section. For parsing, the link scores in (1) are smoothed by word similarities (similar to the approach used by (Wang et al., 2005)) before the maximum score projective dependency tree is computed.

7 Laplacian Regularization

We wish to incorporate similarity based smoothing in large margin training, while using the more refined constraints outlined in Section 5.

Recall that most of the features we use, and therefore most of the parameters we need to estimate are based on bi-lexical parameters $\theta_{ww'}$ that serve as undirected link weights between words w and w' in our dependency parsing model (Section 3). Here we would like to ensure that two different link weights, $\theta_{w_1 w_1'}$ and $\theta_{w_2 w_2'}$, that involve similar words also take on similar values. The previous optimization (7) needs to be modified to take this into account.

Smoothing the link parameters requires us to first extend the notion of word similarity to word-*pair* similarities, since each link involves two words. Given similarities between individual words, computed above, we then define the similarity between word pairs by the geometric mean of the similarities between corresponding words.

$$S(w_1 w_1', w_2 w_2') = \sqrt{S(w_1, w_2) S(w_1', w_2')} \quad (8)$$

where $S(w_1, w_2)$ is defined as in Section 6 above. Then, instead of just solving the constraint system (7) we can also ensure that similar links take on similar parameter values by introducing a penalty on their deviations that is weighted by their similarity value. Specifically, we use

$$\sum_{w_1 w_1'} \sum_{w_2 w_2'} S(w_1 w_1', w_2 w_2') (\theta_{w_1 w_1'} - \theta_{w_2 w_2'})^2 = 2 \boldsymbol{\theta'}^\top L(S) \boldsymbol{\theta'} \quad (9)$$

Here $L(S)$ is the *Laplacian matrix* of S, which is defined by $L(S) = D(S) - S$ where $D(S)$ is a diagonal matrix such that $D_{w_1 w_1', w_1 w_1'} = \sum_{w_2 w_2'} S(w_1 w_1', w_2 w_2')$. Also, $\boldsymbol{\theta'}$ corresponds to the vector of bi-lexical parameters. In this penalty function, if two edges $w_1 w_1'$ and $w_2 w_2'$ have a high similarity value, their parameters will be encouraged to take on similar values. By contrast, if two edges have low similarity, then there will be little mutual attraction on their parameter values.

Note, however, that we do not smooth the parameters, θ_{PMI}, θ_{dist}, θ_{dist2}, corresponding to the pointwise mutual information, distance, and squared distance features described in Section 5, respectively. We only apply similarity smoothing to the bi-lexical parameters.

The Laplacian regularizer (9) provides a natural smoother for the bi-lexical parameter estimates that takes into account valuable word similarity information computed as above. The Laplacian regularizer also has a significant computational advantage: it is guaranteed to be a *convex* quadratic function of the parameters (Zhu et al., 2001). Therefore, by combining the constraint system (7) with the Laplacian smoother (9), we can obtain a convex optimization

Table 1: Accuracy Results on CTB Test Set

Features used	Trained w/ local loss	Trained w/ global loss
Pairs	0.6426	0.6184
+ Lap	0.6506	0.5622
+ Dist	0.6546	0.6466
+ Lap + Dist	0.6586	0.5542
+ MI + Dist	0.6707	0.6546
+ Lap + MI + Dist	0.6827	n/a

Table 2: Accuracy Results on CTB Dev Set

Features used	Trained w/ local loss	Trained w/ global loss
Pairs	0.6130	0.5688
+ Lap	0.6390	0.4935
+ Dist	0.6364	0.6130
+ Lap + Dist	0.6494	0.5299
+ MI + Dist	0.6312	0.6182
+ Lap + MI + Dist	0.6571	n/a

procedure for estimating the link parameters

$$\min_{\boldsymbol{\theta},\boldsymbol{\xi}} \quad \frac{\beta}{2}\boldsymbol{\theta}^\top \tilde{L}(S)\boldsymbol{\theta} + \boldsymbol{\xi}^\top \mathbf{e} \quad \text{subject to} \tag{10}$$
$$\{\boldsymbol{\theta}^\top \mathbf{f}(w_1, w_1') \geq \boldsymbol{\theta}^\top \mathbf{f}(w_1, w_2) + 1 - \xi_{w_1 w_2, w_1 w_1'}\}$$
$$\text{for all constraints in } A$$

where $\tilde{L}(S)$ does not apply smoothing to θ_{PMI}, θ_{dist}, θ_{dist2}.

Clearly, (10) describes a large margin training program for dependency parsing, but one which uses word similarity smoothing for the bi-lexical parameters, and a more refined set of constraints developed in Section 5. Although the constraints are more refined, they are fewer in number than (4). That is, we now only have a polynomial number of constraints corresponding to each word pair in (5), rather than the exponential number over every possible parse tree in (4). Thus, we obtain a polynomial size quadratic program that can be solved for moderately large problems using standard software packages. We used CPLEX in our experiments below. As before, once optimized, the solution parameters $\boldsymbol{\theta}$ can be introduced into the dependency model (1) according to (2).

8 Experimental Results

We tested our method experimentally on the Chinese Treebank (CTB) (Xue et al., 2004). The parse trees in CTB are constituency structures. We converted them into dependency trees using the same method and head-finding rules as in (Bikel, 2004). Following (Bikel, 2004), we used Sections 1-270 for training, Sections 271-300 for testing and Sections 301-325 for development. We experimented with two sets of data: CTB-10 and CTB-15, which contains sentences with no more than 10 and 15 words respectively. Table 1, Table 2 and Table 3 show our experimental results trained and evaluated on Chinese Treebank sentences of length no more than 10, using the standard split. For any unseen link in the new sentences, the weight is computed as the similarity weighted average of similar links seen in the training corpus. The regularization parameter β was set by 5-fold cross-validation on the training set.

Table 3: Accuracy Results on CTB Training Set

Features used	Trained w/ local loss	Trained w/ global loss
Pairs	0.9802	0.8393
+ Lap	0.9777	0.7216
+ Dist	0.9755	0.8376
+ Lap + Dist	0.9747	0.7216
+ MI + Dist	0.9768	0.7985
+ Lap + MI + Dist	0.9738	n/a

We evaluate parsing accuracy by comparing the undirected dependency links in the parser outputs against the undirected links in the treebank. We define the accuracy of the parser to be the percentage of correct dependency links among the total set of dependency links created by the parser.

Table 1 and Table 2 show that training based on the more refined *local loss* is far superior to training with the *global loss* of standard large margin training, on both the test and development sets. Parsing accuracy also appears to increase with the introduction of each new feature. Notably, the pointwise mutual information and distance features significantly improve parsing accuracy—and yet we know of no other research that has investigated these features in this context. Finally, we note that Laplacian regularization improved performance as expected, but not for the *global loss*, where it appears to systematically degrade performance (n/a results did not complete in time). It seems that the global loss model may have been over-regularized (Table 3). However, we have picked the β parameter which gave us the

best resutls in our experiments. One possible explanation for this phenomenon is that the interaction between the Laplician regularization in training and the similarity smoothing in parsing, since distributional word similarities are used in both cases.

Finally, we compared our results to the probabilistic parsing approach of (Wang et al., 2005), which on this data obtained accuracies of 0.7631 on the CTB test set and 0.6104 on the development set. However, we are using a much simpler feature set here.

9 Conclusion

We have presented two improvements to the standard large margin training approach for dependency parsing. To cope with the sparse data problem, we smooth the parameters according to their underlying word similarities by introducing a Laplacian regularizer. More significantly, we use more refined local constraints in the large margin criterion, rather than the global parse-level losses that are commonly considered. We achieve state of the art parsing accuracy for predicting undirected dependencies in test data, competitive with previous large margin and previous probabilistic approaches in our experiments.

Much work remains to be done. One extension is to consider directed features, and contextual features like those used in current probabilistic parsers (Wang et al., 2005). We would also like to apply our approach to parsing English, investigate the confusion showed in Table 3 more carefully, and possibly re-investigate the use of parts-of-speech features in this context.

References

Dan Bikel. 2004. Intricacies of collins' parsing model. *Computational Linguistics*, 30(4).

Eugene Charniak. 2000. A maximum entropy inspired parser. In *Proceedings of NAACL-2000*, pages 132–139.

Colin Cherry and Dekang Lin. 2003. A probability model to improve word alignment. In *Proceedings of ACL-2003*.

M. J. Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proceedings of ACL-1997*.

Aron Culotta and Jeffery Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL-2004*.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *Proceedings of ACL-1999*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING-1996*.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP-2002*.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of EMNLP-2001*, Pittsburgh, PA.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Press, Boston, MA.

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

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. 2004. The entire regularization path for the support vector machine. *JMLR*, 5.

Donald Hindle. 1990. Noun classification from predicate-argument structures. In *Proceedings of ACL-1990*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL-2003*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING/ACL-1998*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL-2005*.

F. Pereira, N. Tishby, and L. Lee. 1993. Distributional clustering of english words. In *Proceedings of ACL-1993*.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3).

B. Taskar, C. Guestrin, and D. Koller. 2003. Max-margin markov networks. In *Proc. of NIPS-2003*.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *Proceedings of EMNLP*.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of ICML-2004*.

Q. Wang, D. Schuurmans, and D. Lin. 2005. Strictly lexical dependency parsing. In *Proceedings of IWPT-2005*.

N. Xue, F. Xia, F. Chiou, and M. Palmer. 2004. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 10(4):1–30.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT-2003*.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000. Unsupervised discovery of scenario-level patterns for information extraction. In *Proceedings of ANLP/NAACL-2000*.

Xiaojin Zhu, John Lafferty, and Zoublin Ghahramani. 2001. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of ICML-2003*.

What are the Productive Units of Natural Language Grammar? A DOP Approach to the Automatic Identification of Constructions.

Willem Zuidema
Institute for Logic, Language and Computation
University of Amsterdam
Plantage Muidergracht 24, 1018 TV, Amsterdam, the Netherlands.
jzuidema@science.uva.nl

Abstract

We explore a novel computational approach to identifying "constructions" or "multi-word expressions" (MWEs) in an annotated corpus. In this approach, MWEs have no special status, but emerge in a general procedure for finding the best statistical grammar to describe the training corpus. The statistical grammar formalism used is that of stochastic tree substitution grammars (STSGs), such as used in Data-Oriented Parsing. We present an algorithm for calculating the expected frequencies of arbitrary subtrees given the parameters of an STSG, and a method for estimating the parameters of an STSG given observed frequencies in a tree bank. We report quantitative results on the ATIS corpus of phrase-structure annotated sentences, and give examples of the MWEs extracted from this corpus.

1 Introduction

Many current theories of language use and acquisition assume that language users store and use much larger fragments of language than the single words and rules of combination of traditional linguistic models. Such fragments are often called constructions, and the theories that assign them a central role "construction grammar" (Goldberg, 1995; Kay and Fillmore, 1999; Tomasello, 2000; Jackendoff, 2002, among others). For construction grammarians, multi-word expressions (MWEs) such as idioms, collocations, fixed expressions and compound verbs and nouns, are not so much exceptions to the rule, but rather extreme cases that reveal some fundamental properties of natural language.

In the construction grammar tradition, co-occurrence statistics from corpora have often been used as evidence for hypothesized constructions. However, such statistics are typically gathered on a case-by-case basis, and no reliable procedure exists to automatically identify constructions. In contrast, in computational linguistics, many automatic procedures are studied for identifying MWEs (Sag et al., 2002) – with varying success – but here they are treated as exceptions: identifying multi-word expressions is a pre-processing step, where typically adjacent words are grouped together after which the usual procedures for syntactic or semantic analysis can be applied. In this paper I explore an alternative formal and computational approach, where multi-word constructions have no special status, but emerge in a general procedure to find the best statistical grammar to describe a training corpus. Crucially, I use a formalism known as "Stochastic Tree Substitution Grammars" (henceforth, STSGs), which can represent single words, contiguous and noncontiguous MWEs, context-free rules or complete parse trees in a unified representation.

My approach is closely related to work in statistical parsing known as Data-Oriented Parsing (DOP), an empirically highly successful approach with labeled recall and precision scores on the Penn Tree Bank that are among the best currently obtained (Bod, 2003). DOP, first proposed in (Scha, 1990),

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 29–36, New York City, June 2006.

can be seen as an early formalization and combination of ideas from construction grammar and statistical parsing. Its key innovations were (i) the proposal to use fragments of trees from a tree bank as the symbolic backbone; (ii) the proposal to allow, in principle, trees of arbitrary size and shape as the elementary units of combination; (iii) the proposal to use the occurrence and co-occurrence frequencies as the basis for structural disambiguation in parsing.

The model I develop in this paper is true to these general DOP ideals, although it differs in important respects from the many DOP implementations that have been studied since its first inception (Bod, 1993; Goodman, 1996; Bod, 1998; Sima'an, 2002; Collins and Duffy, 2002; Bod et al., 2003, and many others). The crucial difference is in the estimation procedure for choosing the weights of the STSG based on observed frequencies in a corpus. Existing DOP models converge to STSGs that either (i) give all subtrees of the observed trees nonzero weights (Bod, 1993; Bod, 2003), or (ii) give only the largest possible fragments nonzero weights (Sima'an and Buratto, 2003; Zollmann and Sima'an, 2005). The model in this paper, in contrast, aims at finding the smallest set of productive units that explain the occurrences and co-occurrences in a corpus. Large subtrees only receive non-zero weights, if they occur more frequently than can be expected on the basis of the weights of smaller subtrees.

2 Formalism, Notation and Definitions

2.1 Stochastic Tree Substitution Grammars

STSGs are a simple generalization of Stochastic Context Free Grammars (henceforth, SCFGs), where the productive units are elementary trees of arbitrary size instead of the rewrite rules in SCFGs (which can be viewed as trees of depth 1). STSGs form a restricted subclass of Stochastic Tree Adjoining Grammars (henceforth, STAGs) (Resnik, 1992; Schabes, 1992), the difference being that STSGs only allow for substitution and not for adjunction (Joshi and Sarkar, 2003). This limits the generative capacity to that of context-free grammars, and means STSGs cannot be fully lexicalized. These limitations notwithstanding, the close relationship with STAGs is an attractive feature with extensions to the class of mildly context-sensitive languages (Joshi et al., 1991) in mind. Most importantly, however, STSGs are already able to model a vast range of statistical dependencies between words and constituents, which allows them to rightly predict the occurrences of many constructions (Bod, 1998).

For completeness, we include the usual definitions of STSGs, the substitution operation and derivation and parse probabilities (Bod, 1998), using our own notation. An STSG is a 5-tuple $\langle V_n, V_t, S, T, w\rangle$, where V_n is the set of non-terminal symbols; V_t is the set of terminal symbols; $S \in V_n$ is the start symbol; T is a set of elementary trees, such that for every $t \in T$ the unique root node $r(t) \in V_n$, the set of internal nodes $i(t) \subset V_n$ and the set of leaf nodes $l(t) \subset V_n \cup V_t$; finally, $w : T \rightarrow [0, 1]$ is a probability (weight) distribution over the elementary trees, such that for any $t \in T$, $\sum_{t' \in R(t)} w(t') = 1$, where $R(t)$ is the set of elementary trees with the same root label as t. It will prove useful to also define the set of all possible trees θ over the defined alphabets (with the same conditions on root, internal and leaf nodes as for T), and the set of all possible *complete* parse trees Θ (with $r(t) = S$ and all leaf nodes $l(t) \subset V_t$). Obviously, $T \subset \theta$ and $\Theta \subset \theta$.

The substitution operation $\circ$ is defined if the *leftmost nonterminal leaf* in t_1 is identical to the root of t_2. Performing substitution $t_1 \circ t_2$ yields t_3, if t_3 is identical to t_1 with the leftmost nonterminal leaf replaced by t_2. A derivation is a sequence of elementary trees, where the first tree $t \in T$ has root-label S and every next tree combines through substitution with the result of the substitutions before it. The probability of a derivation d is defined as the product of weights of the elementary trees involved:

$$P(d = t_1 \circ \ldots \circ t_n) = \prod_{i=1}^{n} \left(w\left(t_i\right)\right). \qquad (1)$$

A *parse tree* is any tree $t \in \Theta$. Multiple derivations can yield the same parse tree; the probability of a parse tree p equals the sum of the probabilities of the different derivations that yield that same tree:

$$P(p) \;=\; \sum_{d:\hat{d}=p} \left(P\left(d\right)\right), \qquad (2)$$

where $\hat{d}$ is the tree derived by derivation d.

In this paper, we are only concerned with grammars that define proper probability distributions over

trees, such that the probability of all derivations sum up to 1 and no probability mass gets lost in derivations that never reach a terminal yield. We require:

$$\sum_{p \in \Theta} P(p) = \sum_{d : \hat{d} \in \Theta} P(d) = 1. \quad (3)$$

2.2 Usage Frequency and Occurrence Frequency

In addition to these conventional definitions, we will make use in this paper of the concepts "usage frequency" and "occurrence frequency". When we consider an arbitrary subtree t, the usage frequency $u(t)$ describes the relative frequency with which elementary tree t is involved in a set of derivations. Given a grammar $G \in$ STSG, the expected usage frequency is:

$$u(t) = \sum_{d : t \in d} \left(P(d)\, C(t, d) \right), \quad (4)$$

where $C(t, d)$ gives the number of occurrences of t in d. The set of derivations, and hence usage frequency, is usually considered hidden information.

The occurrence frequency $f(t)$ describes the relative frequency with which t occurs as a subtree of a set of parse trees, which is usually assumed to be observable information. If grammar G is used to generate trees, it will create a tree bank where each parse tree will occur with an expected frequency as in equation (2). More generally, the expected occurrence frequency $f(t)$ (relative to the number n of complete trees in the tree bank) of a subtree t is:

$$\mathbf{E}[f(t)] = \sum_{p : t \in p^*} \left(P(p)\, C(t, p^*) \right), \quad (5)$$

where p^* is the multiset of all subtrees of p.

Hence, $w(t)$, $u(t)$ and $f(t)$ all assign values (the latter two not necessarily between 0 and 1) to trees. An important question is how these different values can be related. For STSGs which have only elementary trees of depth 1, and are thus equivalent to SCFGs, these relations are straightforward: the usage frequency of an elementary tree simply equals its expected frequency, and can be derived from the weights by multiplying inside and outside probabilities (Lari and Young, 1990). Estimating the weights of an (unconstrained and untransformed) SCFG from an tree bank is straightforward, as weights, in the limit, simply equal the relative frequency of each depth-1 subtree (relative to other depth-1 subtrees with the same root label).

When elementary trees can be of arbitrary depth, however, many different derivations can yield the same tree, and a given subtree t can emerge without the corresponding elementary tree ever having been used. The expected frequencies are sums of products, and – if one wants to avoid exhaustively enumerating all possible parse trees – surprisingly difficult to calculate, as will become clear below.

2.3 From weights to usage frequencies and back

Relating usage frequencies to weights is relatively simple. With a bit of algebra we can work out the following relations:

$$u(t) = \begin{cases} w(t) & \text{if } r(t) = S \\ w(t) \sum_{t' : r(t) \in l(t')} u(t') C_t^{t'} & \text{otherwise} \end{cases} \quad (6)$$

where $C_t^{t'}$ gives the number of occurrences of the root label $r(t)$ of t among the leaves of t'. The inverse relation is straightforward:

$$w(t) = \frac{u(t)}{\sum_{t' \in R(t)} u(t')}. \quad (7)$$

2.4 From usage frequency to expected frequency

The two remaining problems – calculating expected frequencies from weights and estimating the weights from observed frequencies – are surprisingly difficult and heretofore not satisfactorily solved. In (Zuidema, 2006) we evaluate existing estimation methods for Data-Oriented Parsing, and show that they are ill-suited for learning tasks such as studied in this paper. In the next section, we present a new algorithm for estimation, which makes use of a method for calculating expected frequencies that we sketch in this section. This method makes use of sub- and supertree relations that we explain first.

We define two types of subtrees of a given tree t, which, for lack of better terminology, we will call "twigs" and "prunes" of t. Twigs are those subtrees headed by any of t's internal nodes and everything

below. Prunes are those subtrees headed by t's root-node, pruned at any number (≥ 0) of internal nodes. Using $\circ$ to indicate left-most substitution, we write:

- t_1 is a *twig* of t_2, if either $t_1 = t_2$ or $\exists t_3$, such that $t_3 \circ t_1 = t_2$;
- t_1 is a *prune* of t_2, if either $t_1 = t_2$ or $\exists t_3 \ldots t_n$, such that $t_1 \circ t_3 \ldots \circ t_n = t_2$;
- $t' = pr_x(t)$, if x is a set of nodes in t, such that if t is pruned at each $i \in x$ it equals t'.

Thus defined, the set of all subtrees $st(t)$ of t corresponds to the set of all prunes of all twigs of t: $st(t) = \{t''|\exists t'(t' \in tw(t) \wedge t'' \in pr(t'))$.

We further define the sets of supertwigs, superprunes and supertrees as follows:

- $\widehat{tw}(t) = \{t'|t \in tw(t')\}$
- $\widehat{pr_x}(t) = \{t'|t = pr_x(t')\}$
- $\widehat{st}(t) = \{t'|t \in st(t')\}$.

Using these sets, and the set of derivations $D(t)$ of the fragment t, a general expression for the expected frequency of t is:

$$\begin{aligned} \mathbf{E}[f(t)] &= \sum_{d \in D(t)} \alpha\beta \\ \alpha &= \sum_{\tau \in \widehat{tw}(d_1)} \sum_{\tau' \in \widehat{pr_{x(t)}}(\tau)} u(\tau') \\ \beta &= \prod_{\substack{t' \in \\ \langle d_2, \ldots, d_n \rangle}} \sum_{\tau' \in \widehat{pr_{x(t)}}(t')} w\left(\tau'\right) \end{aligned} \quad (8)$$

where $\langle d_1, \ldots, d_n \rangle$ is the sequence of elementary trees in derivation d. A derivation of this equation is provided on the author's website[1]. Note that it will, in general, be computationally extremely expensive to calculate $\mathbf{E}[f(t)]$. We will come back to computational efficiency issues in the discussion.

[1] http://staff.science.uva.nl/∼jzuidema. The intuition behind it is as follows. Observe first that there are many ways in which an arbitrary fragment t can emerge, many of which do not involve the usage of the *elementary tree* t. It is useful to partition the set of all derivations of complete parse trees according to the substitution sites inside t that they involve, and hence according to the corresponding derivations of t. The first summation in (8) simply sums over all these cases.

Each derivation of t involves a first elementary tree d_1, and possibly a sequence of further elementary trees $\langle d_2, \ldots, d_n \rangle$. Roughly speaking, the α-term in equation (8) describes the frequency with which a d_1 will be generated. The β-term then describes the probability that d_1 will be expanded as t. The equation simplifies considerably for those fragments that have no nonterminal leaves: the set $\widehat{pr_{x(t)}}$ then only contains t, and the two summations over this set disappear. The equation further simplifies if only depth-1 elementary trees have nonzero weights (i.e. for SCFGs): α and β then essentially give outside and inside probabilities (Lari and Young, 1990). However, for unconstrained STSGs we need all sums and products in (8).

3 Estimation: push-n-pull

The goal of this paper is an automatic discovery procedure for finding "constructions" based on occurrence and co-occurrence frequencies in a corpus. Now that we have introduced the necessary terminology, we can reformulate this goal as follows: *What are the elementary trees with multiple words with the highest usage frequency in the STSG estimated from an annotated corpus?* Thus phrased, the crucial next step is to decide on an estimation procedure for learning an STSG from a corpus.

Here we develop an estimation procedure we call "push-n-pull". The basic idea is as follows. Given an initial setting of the parameters, the method calculates the expected frequency of all complete and incomplete trees. If a tree's expected frequency is higher than its observed frequency, the method subtracts the difference from the tree's score, and distributes ("pushes") it over the trees involved in its derivations. If it is lower, it "pulls" the difference from these same derivations. The method includes a bias for moving probability mass to smaller elementary trees, to avoid overfitting; its effects become smaller as more data gets observed.

Because the method for calculating estimated frequency works with usage-frequencies, the push-n-pull algorithm also uses these as parameters. More precisely, it manipulates a "score", which is the product of usage frequency and the total number of parse trees observed. Implicit here is the assumption that by shifting usage frequencies between different derivations, the relation with weights remains as in equation (6). Simulations suggest this is reasonable.

In the current implementation, the method starts with all frequency mass in the longest derivations, i.e. in the depth-1 elementary trees. Finally, the current implementation is incremental. It keeps track of the frequencies with which it observes subtrees in a corpus. For each tree received, it finds all derivations and all probabilities, updates frequencies and scores according to the rules sketched above. In pseudo-code, the push-n-pull algorithm is as follows:

for each observed parse tree p

for each depth-1 subtree t in p
    update-score$(t, 1.0)$
for each subtree t of p
    $\Delta =$min$(sc(t), B + \gamma(\mathbf{E}[f(t)] - f(t)))$
    $\Delta' = 0$
    for each of n derivations d of t
        let $t' \ldots t''$ be all elementary trees in d
            $\delta =$min$(sc(t'), \ldots, sc(t''), -\Delta/n)$
        $\Delta' -= \delta$
        for each elementary tree t' in d
            update-score(t', δ)
    update-score (t, Δ')

where $sc(t)$ is the score of t, B is the bias towards smaller subtrees, γ is the learning rate parameter and $f(t)$ is the observed frequency of t. Δ' thus gives the actual change in the score of t, based on the difference between expected and observed frequency, bias, learning rate and how much scores can be pushed or pulled[2]. For computational efficiency, only subtrees with a depth no larger than $d = 3$ or $d = 4$ and only derivations involving 2 elementary trees are considered.

4 Results

We have implemented the algorithms for calculating the expected frequency, and the push-n-pull algorithm for estimation. We have evaluated the algorithms on a number of simple example STSGs and found that the expected frequency algorithm correctly predicts observed frequencies. We have further found that – unlike existing estimation methods – the push-n-pull algorithm converges to STSGs that closely model the observed frequencies (i.e. that maximize the likelihood of the data) without putting all probability mass in the largest elementary trees (i.e. whilst retaining generalizations about the data).

Here we report first quantitative results on the ATIS3 corpus (Hemphill et al., 1990). Before processing, all trees (train and test set) were converted to a format that our current implementation requires (all non-terminal labels are unique, all internal nodes have two daughters, all preterminal nodes have a single lexical daughter; all unary productions and all traces were removed). The set of trees was randomly split in a train set of 462 trees, and a test set of 116 trees. The push-n-pull algorithm was then run in 10 passes over the train set, with $d = 3$, $B = 0$ and $\gamma = 0.1$. By calculating the most probable parse[3] for each yield of the trees in test set, and running "evalb" we arrive at the following quantitative results: a string set coverage of 84% (19 failed parses), labeled recall of 95.07, and labeled precision of 95.07. We obtained almost identical numbers on the same data with a reimplementation of the DOP1 algorithm (Bod, 1998).

method	# rules	Cov.	LR	LP	EM
DOP1	77852	84%	95.07	95.07	83.5
p-n-p	58799	84%	95.07	95.07	83.5

Table 1: Parseval scores of DOP1 and push-n-pull on the same 462-116 random train-testset split of a treebank derived from the ATIS3 corpus (we emphasize that all trees, also those of the test-set, were converted to Chomsky Normal Form, whereby unary production and traces were removed and top-nodes relabeled "TOP". These results are thus not comparable to previous methods evaluated on the ATIS3 corpus.) EM is "exact match".

method	# rules	Cov.	LR	LP	EM
$sc > 0.3$	8593	77%	80.8	80.8	46.3
$sc > 0.1$	98443	77%	81.9	81.9	48.8

Table 2: Parseval scores using a p-n-p induced STSG on the same treebank as in table 1, using a different random 525-53 train-testset split. Shown are results were only elementary trees with scores higher than 0.3 and 0.1 respectively are used.

However, more interesting is a qualitative analysis of the STSG induced, which shows that, unlike DOP1, push-n-pull arrives at a grammar that gives high weights (and scores) to those elementary

[2] An important topic for future research is to clarify the relation between push-n-pull and Expectation Maximization.

[3] We approximated the most probable parse as follows (following (Bod, 2003)). We first converted the induced STSG to an isomorph SCFG, by giving the internal nodes of every elementary tree t unique address-labels, and reading off all CFG productions (all with weight 1.0, except for the top-production, which receives the weight of t). An existing SCFG parser (Schmid, 2004) was then used, with a simple unknown word heuristic, to generate the Viterbi n-best parses with $n = 100$, and, after removing the address labels, all equal parses and their probabilities were summed, and the one with highest probability chosen.

trees that best explain the overrepresentation of certain constructions in the data. For instance, in a run with $d = 4, \gamma = 1.0, B = 1.0$, the 50 elementary trees with the highest scores, as shown in figure 1, are all exemplary of frequent formulas in the ATIS corpus such as "show me X", "I'd like to X", "which of these", "what is the X", "cheapest fare" and "flights from X to Y". In short, the push-n-pull algorithm – while starting out considering all possible subtrees – converges to a grammar which makes linguistically relevant generalizations. This allows for a more compact grammar (58799 rules in the SCFG reduction, vs. 77852 for DOP1), whilst retaining DOP's excellent empirical performance.

5 Discussion

Calculating $\mathbf{E}[f(t)]$ using equation (8) can be extremely expensive in computational terms. One will typically want to calculate this value for all subtrees, the number of which is exponential in the size of the trees in the training data. For each subtree t, we will need to consider the set of all its derivations (exponential in the size of t), and for each derivation the set of supertwigs of the first elementary trees and, for incompletely lexicalized subtrees, the set of superprunes of all elementary trees in their derivations. The latter two sets, however, need not be constructed for every time the *expected* frequency $\mathbf{E}[f(t)]$ is calculated. Instead, we can, as we do in the current implementation, keep track of the two sums for every change of the weights.

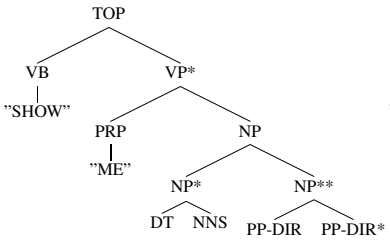
However, there are many further possibilities for improving the efficiency of the algorithm that are currently not implemented. Equation (8) remains valid under various restrictions on the elementary trees that we are willing to consider as productive units. Some of these will remove the exponential dependence on the size of the trees in the training data. For instance, in the case where we restrict the productive units (with nonzero weights) to depth-1 trees (i.e. CFG rules), equation (8) collapses to the product of inside and outside probabilities, which can be calculated using dynamical programming in polynomial time (Lari and Young, 1990). A major topic for future research is to define linguistically motivated restrictions that allow for efficient computation.

Another concern is the size of the grammar the estimation procedure produces, and hence the time and space efficiency of the resulting parser. Table 1 already showed that push-n-pull leads to a more concise grammar. The reason is that many potential elementary trees receive a score (and weight) 0. More generally, push-n-pull generates extremely tilted score distributions, which allows for even more compact but highly accurate approximations. In table 2 we show, for the $d = 4$ grammar of figure 1, that a 10-fold reduction of the grammar size by pruning elementary trees with low scores, leads only to a small decrease in the LP and LR measures.

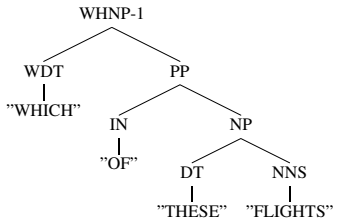
Another interesting question is if and how the current algorithm can be extended to the full class of Stochastic Tree-Adjoining Grammars (Schabes, 1992; Resnik, 1992). With the added operation of adjunction, equation (8) is not valid anymore. Given the computational complexities that it already gives rise to, however, it seems that issue of linguistically motivated restrictions (other than lexicalization) should be considered first. Finally, given that the current approach is dependent on the availability of a large annotated corpus, an important question is if and how it can be extended to work with unlabeled data. That is, can we transform the push-n-pull algorithm to perform the unsupervised learning of STSGs? Although most work on unsupervised grammar learning concerns SCFGs (including some of our own (Zuidema, 2003)) it is interesting to note that much of the evidence for construction grammar in fact comes from the language acquisition literature (Tomasello, 2000).

6 Conclusions

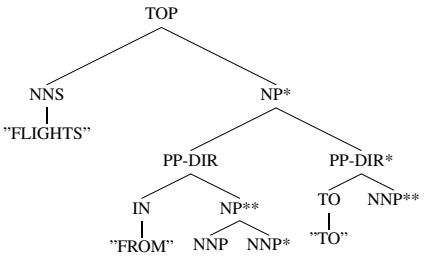
Theoretical linguistics has long strived to account for the unbounded productivity of natural language syntax with as few units and rules of combination as possible. In contrast, construction grammar and related theories of grammar postulate a heterogeneous and redundant storage of "constructions". If this view is correct, we expect to see statistical signatures of these constructions in the distributional information that can be derived from corpora of natural language utterances. How can we recover those signatures? In this paper we have presented an approach to identifying the relevant statistical correlations in a corpus based on the assumption that the



(a) The "show me NP PP" frame, which occurs very frequently in the training data and is represented in several elementary trees with high weight.



(b) The complete parse tree for the sentence "Which of these flights", which occurs 16 times in training data.



(c) The frame for "flights from NP to NP"

```
1.  ((TOP (VB "SHOW") (VP* (PRP "ME") (NP (NP* DT NNS) (NP** PP-DIR PP-DIR*)))) 17.79 0.008 30)
2.  ((TOP (VB "SHOW") (VP* (PRP "ME") (NP (NP* DT NNS) NP**))) 10.34 0.004 46
3.  (TOP (PRP "I") (VP (MD "WOULD") (VP* (VB "LIKE") (VP** TO VP***)))) 10.02 0.009 20
4.  (WHNP-1 (WDT "WHICH") (PP (IN "OF") (NP (DT "THESE") (NNS "FLIGHTS")))) 8.80 0.078 16
5.  (TOP (WP "WHAT") (SQ (VBZ "IS") (NP-SBJ (DT "THE") (NN "PRICE")))) 8.76 0.005 20
6.  (TOP (WHNP (WDT "WHAT") (NNS "FLIGHTS")) (SQ (VBP "ARE") (SQ* (EX "THERE") SQ**))) 8.25 0.006 36
7.  (VP* (PRP "ME") (NP (NP* (DT "THE") (NNS "FLIGHTS")) (NP** (PP-DIR IN NNP) (PP-DIR* TO NNP*)))) 7.90 0.023 18
8.  (TOP (WHNP (WDT "WHAT") (NNS "FLIGHTS")) (SQ (VBP "ARE") (SQ* (EX "THERE") (SQ** PP-DIR-3 PP-DIR-4)))) 6.64 0.005 26
9.  (TOP (PRP "I") (VP MD (VP* (VB "LIKE") (VP** TO VP***)))) 6.48 0.006 20
10. (TOP (PRP "I") (VP (VBP "NEED") (NP (NP* DT NN) (NP** PP-DIR NP***)))) 5.01 0.004 10
11. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (DT "THE") NNS))) 4.94 0.002 16
12. (TOP WP (SQ (VBZ "IS") (NP-SBJ (DT "THE") (NN "PRICE")))) 4.91 0.0028 20
13. (TOP (WHNP (WDT "WHAT") (NNS "FLIGHTS")) (SQ (VBP "ARE") (SQ* EX (SQ** PP-DIR-3 PP-DIR-4)))) 4.16 0.003 26
14. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (NNS "FLIGHTS") NP*))) 4.01 0.001 16
15. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (DT "THE") NP*))) 3.94 0.002 12
16. (TOP (WHNP (WDT "WHAT") (NNS "FLIGHTS")) (SQ (VBP "ARE") (SQ* EX SQ**))) 3.92 0.003 36
17. (TOP (PRP "I") (VP (VBP "NEED") (NP (NP* DT NN) NP**)))) 3.85 0.003 14
18. (TOP (WP "WHAT") (SQ VBZ (NP-SBJ (DT "THE") (NN "PRICE")))) 3.79 0.002 20
19. (WHNP-1 (WDT "WHICH") (PP (IN "OF") (NP (DT "THESE") NNS))) 3.65 0.032 16
20. (TOP (VB "SHOW") (VP* (PRP "ME") (NP NP* (SBAR WDT VP**)))) 3.64 0.002 14
21. (TOP (VB "SHOW") (VP* PRP (NP (NP* DT NNS) (NP** PP-DIR PP-DIR*)))) 3.61 0.002 30
22. (TOP (WHNP (WDT "WHAT") NNS) (SQ (VBP "ARE") (SQ* (EX "THERE") (SQ** PP-DIR-3 PP-DIR-4)))) 3.30 0.002 26
23. (VP (MD "WOULD") (VP* (VB "LIKE") (VP** (TO "TO") (VP*** VB* VP****)))) 3.25 0.012 16
24. (TOP (WDT "WHICH") VP) 3.1460636 0.001646589 12
25. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (NP* DT NP**) NP***))) 3.03 0.001 12
26. (TOP (VB "SHOW") (VP* (PRP "ME") (NP NP* (NP*** PP-DIR PP-DIR*)))) 2.97 0.001 12
27. (PP (IN "OF") (NP* (NN* "FLIGHT") (NP** NNP (NP*** NNP* NP****)))) 2.95 0.015 8
28. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (DT "THE") (NNS "FARES")))) 2.85 0.001 8
29. (VP (VBP "NEED") (NP (NP* (DT "A") (NN "FLIGHT")) (NP** PP-DIR NP***))) 2.77 0.009 12
30. (TOP (VB "SHOW") (VP* (PRP "ME") (NP NP* (NP** PP-DIR PP-DIR*)))) 2.77 0.001 34
31. (TOP (JJS "CHEAPEST") (NN "FARE")) 2.74 0.001 6
32. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (NP* DT NP**) (NP*** PP-DIR PP-DIR*)))) 2.71 0.001 8
33. (TOP (NN "PRICE") (PP (IN "OF") (NP* (NN* "FLIGHT") (NP** NNP NP***)))) 2.69 0.001 6
34. (TOP (NN "PRICE") (PP (IN "OF") (NP* (NN* "FLIGHT") NP**))) 2.68 0.001 8
35. (PP-DIR (IN "FROM") (NP (NNP "WASHINGTON") (NP* (NNP* "D") (NNP** "C")))) 2.67 0.006 6
36. (PP-DIR (IN "FROM") (NP** (NNP "NEWARK") (NP*** (NNP* "NEW") (NNP** "JERSEY")))) 2.60 0.005 6
37. (S* (PRP "I") (VP (MD "WOULD") (VP* (VB "LIKE") (VP** TO VP***)))) 2.59 0.11 8
38. (TOP (VBZ "DOES") (SQ* (NP-SBJ DT (NN "FLIGHT")) (VP (VB "SERVE") (NN* "DINNER")))) 2.48 0.002 8
39. (TOP (PRP "I") (VP (MD "WOULD") (VP* (VB "LIKE") VP**))) 2.37 0.002 20
40. (TOP (WP "WHAT") (SQ (VBZ "IS") (NP-SBJ DT (NN "PRICE")))) 2.33 0.001 20
41. (S* (PRP "I") (VP MD (VP* (VB "LIKE") (VP** TO VP***)))) 2.33 0.100 8
42. (WHNP**** (PP-TMP (IN* "ON") (NNP** "FRIDAY")) (PP-LOC (IN** "ON") (NP (NNP*** "AMERICAN") (NNP**** "AIRLINES")))) 2.30 0.086 6
43. (VP* (PRP "ME") (NP (NP* (DT "THE") NNS) (NP** (PP-DIR IN NNP) (PP-DIR* TO NNP*)))) 2.29 0.007 18
44. (TOP (WHNP* (WDT "WHAT") (NNS "FLIGHTS")) (WHNP** (PP-DIR (IN "FROM") NNP) (WHNP*** (PP-DIR* TO NNP*) (PP-TMP IN* NNP**)))) 2.28 0.001 12
45. (SQ (VBP "ARE") (SQ* EX (SQ** (PP-DIR-3 IN NNP) (PP-DIR-4 TO NNP*)))) 2.26 0.015 14
46. (TOP (VB "SHOW") (VP* (PRP "ME") (NP (NP* DT NNS) (SBAR WDT VP**)))) 2.22 0.001 8
47. (TOP (NNS "FLIGHTS") (NP* (PP-DIR (IN "FROM") (NP** NNP NNP*)) (PP-DIR* (TO "TO") NNP**))) 2.20 0.001 10)
48. ((VP (VBP "NEED") (NP (NP* (DT "A") (NN "FLIGHT")) (NP** (PP-DIR IN NNP) NP***))) 2.1346128 0.007185978 10)
49. ((NP (NP* (DT "THE") (NNS "FLIGHTS")) (NP** (PP-DIR (IN "FROM") (NNP "BALTIMORE")) (PP-DIR* (TO "TO") (NNP* "OAKLAND")))) 2.1335514 0.00381956 10)
50. ((TOP (VB "SHOW") (VP* (PRP "ME") (NP (NP* DT NNS) (NP** PP-DIR NP***)))) 2.09 0.001 8)
```

Figure 1: Three examples and a list of the first 50 elementary trees with multiple words of an STSG induced using the push-n-pull algorithm on the ATIS3 corpus. For use in the current implementation, the parse trees have been converted to Chomsky Normal Form (all occurrences of $A \rightarrow B$, $B \rightarrow \omega$ are replaced by $A \rightarrow \omega$; all occurrences of $A \rightarrow BC\omega$ are replaced by $A \rightarrow BA*$, $A* \rightarrow C\omega$), all non-terminal labels are made unique for a particular parse tree (address labeling not shown) and all top nodes are replaced by the non-terminal "TOP". Listed are the elementary trees of the induced STSG with for each tree the score, the weight and the frequency with which it occurs in the training set.

corpus is generated by an STSG, and by inferring the properties of that underlying STSG. Given our best guess of the STSG that generated the data, we can start to ask questions like: which subtrees are overrepresented in the corpus? Which correlations are so strong that it is reasonable to think of the correlated phrases as a single unit? We presented a new algorithm for estimating weights of an STSG from a corpus, and reported promising empirical results on a small corpus.

Acknowledgments

The author is funded by the Netherlands Organisation for Scientific Research (Exacte Wetenschappen), project number 612.066.405. Many thanks to Yoav Seginer, Rens Bod and Remko Scha and the anonymous reviewers for very useful comments.

References

Rens Bod, Remko Scha, and Khalil Sima'an, editors. 2003. *Data-Oriented Parsing*. CSLI Publications, University of Chicago Press, Chicago, IL.

Rens Bod. 1993. Using an annotated corpus as a stochastic grammar. In *Proceedings EACL'93*, pages 37–44.

Rens Bod. 1998. *Beyond Grammar: An experience-based theory of language*. CSLI, Stanford, CA.

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings EACL'03*.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. *ACL'02*.

Adele E. Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. The University of Chicago Press, Chicago, IL.

Joshua Goodman. 1996. Efficient algorithms for parsing the DOP model. In *Proceedings EMNLP'96*, p. 143–152.

C.T. Hemphill, J.J. Godfrey, and G.R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA Speech and Natural Language Workshop*. Morgan Kaufman, Hidden Valley.

Ray Jackendoff. 2002. *Foundations of Language*. Oxford University Press, Oxford, UK.

Aravind Joshi and Anoop Sarkar. 2003. Tree adjoining grammars and their application to statistical parsing. In Bod et al. (Bod et al., 2003), pages 253–282.

A. Joshi, K. Vijay-Shanker, and D. Weir. 1991. The convergence of mildly context-sensitive grammar formalisms. In Peter Sells, Stuart Shieber, and Tom Wasow, editors, *Foundational issues in natural language processing*, pages 21–82. MIT Press, Cambridge MA.

P. Kay and C. Fillmore. 1999. Grammatical constructions and linguistic generalizations. *Language*, 75:1–33.

K. Lari and S.J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

Philip Resnik. 1992. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings COLING'92*, p. 418–424.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann A. Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings CICLing*, pages 1–15.

Remko Scha. 1990. Taaltheorie en taaltechnologie; competence en performance. In R. de Kort and G.L.J. Leerdam, editors, *Computertoepassingen in de Neerlandistiek*, pages 7–22. LVVN, Almere. http://iaaa.nl/rs/LeerdamE.html.

Yves Schabes. 1992. Stochastic lexicalized tree-adjoining grammars. In *Proceedings COLING'92*, pages 425–432.

Helmut Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings COLING'04*.

Khalil Sima'an and Luciano Buratto. 2003. Backoff parameter estimation for the DOP model. In *Proceedings ECML'03*, pages 373–384.

Khalil Sima'an. 2002. Computational complexity of probabilistic disambiguation. *Grammars*, 5(2):125–151.

Michael Tomasello. 2000. The item-based nature of children's early syntactic development. *Trends in Cognitive Science*, 4(4):156–163.

Andreas Zollmann and Khalil Sima'an. 2005. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*.

Willem Zuidema. 2003. How the poverty of the stimulus solves the poverty of the stimulus. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 51–58. MIT Press, Cambridge, MA.

Willem Zuidema. 2006. Theoretical evaluation of estimation methods for Data-Oriented Parsing. In *Proceedings EACL'06 (Conference Companion)*, pages 183–186.

Resolving and Generating Definite Anaphora by Modeling Hypernymy using Unlabeled Corpora

Nikesh Garera and **David Yarowsky**
Department of Computer Science
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218, USA
{ngarera,yarowsky}@cs.jhu.edu

Abstract

We demonstrate an original and successful approach for both resolving and generating definite anaphora. We propose and evaluate unsupervised models for extracting hypernym relations by mining cooccurrence data of definite NPs and potential antecedents in an unlabeled corpus. The algorithm outperforms a standard WordNet-based approach to resolving and generating definite anaphora. It also substantially outperforms recent related work using pattern-based extraction of such hypernym relations for coreference resolution.

1 Introduction

Successful resolution and generation of definite anaphora requires knowledge of hypernym and hyponym relationships. For example, determining the antecedent to the definite anaphor "*the drug*" in text requires knowledge of what previous noun-phrase candidates could be drugs. Likewise, generating a definite anaphor for the antecedent "*Morphine*" in text requires both knowledge of potential hypernyms (e.g. "*the opiate*", "*the narcotic*", "*the drug*", and "*the substance*"), as well as selection of the most appropriate level of generality along the hypernym tree in context (i.e. the "natural" hypernym anaphor). Unfortunately existing manual hypernym databases such as WordNet are very incomplete, especially for technical vocabulary and proper names. WordNets are also limited or non-existent for most of the world's languages. Finally, WordNets also do not include notation of the "natural" hypernym level for anaphora generation, and using the immediate parent performs quite poorly, as quantified in Section 5. In first part of this paper, we propose a novel approach for *resolving* definite anaphora involving hyponymy relations. We show that it performs substantially better than previous approaches on the task of antecedent selection. In the second part we demonstrate how this approach can be successfully extended to the problem of *generating* a natural definite NP given a specific antecedent.

In order to explain the antecedent selection task for definite anaphora clearly, we provide the following example taken from the LDC Gigaword corpus (Graff *et al.*, 2005).

(1)...*pseudoephedrine* is found in an allergy treatment, which was given to Wilson by a doctor when he attended Blinn junior college in Houston. In a unanimous vote, the Norwegian sports confederation ruled that Wilson had not taken **the drug** to enhance his performance...

In the above example, the task is to resolve the definite NP *the drug* to its correct antecedent *pseudoephedrine*, among the potential antecedents *<pseudoephedrine, allergy, blinn, college, houston, vote, confederation, wilson>*. Only *Wilson* can be ruled out on syntactic grounds (Hobbs, 1978). To be able to resolve the correct antecedent from the remaining potential antecedents, the system requires the knowledge that *pseudoephedrine* is a *drug*. Thus, the problem is to create such a knowledge source and apply it to this task of antecedent selection. A total of 177 such anaphoric examples

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 37–44, New York City, June 2006.

were extracted randomly from the LDC Gigaword corpus and a human judge identified the correct antecedent for the definite NP in each example (given a context of previous sentences).[1] Two human judges were asked to perform the same task over the same examples. The agreement between the judges was 92% (of all 177 examples), indicating a clearly defined task for our evaluation purposes.

We describe an unsupervised approach to this task that extracts examples containing definite NPs from a large corpus, considers all head words appearing before the definite NP as potential antecedents and then filters the noisy <antecedent, definite-NP> pair using Mutual Information space. The co-occurence statistics of such pairs can then be used as a mechanism for detecting a hypernym relation between the definite NP and its potential antecedents. We compare this approach with a WordNet-based algorithm and with an approach presented by Markert and Nissim (2005) on resolving definite NP coreference that makes use of lexico-syntactic patterns such as *'X and Other Ys'* as utilized by Hearst (1992).

2 Related work

There is a rich tradition of work using lexical and semantic resources for anaphora and coreference resolution. Several researchers have used WordNet as a lexical and semantic resource for certain types of bridging anaphora (Poesio *et al.*, 1997; Meyer and Dale, 2002). WordNet has also been used as an important feature in machine learning of coreference resolution using supervised training data (Soon *et al.*, 2001; Ng and Cardie, 2002). However, several researchers have reported that knowledge incorporated via WordNet is still insufficient for definite anaphora resolution. And of course, WordNet is not available for all languages and is missing inclusion of large segments of the vocabulary even for covered languages. Hence researchers have investigated use of corpus-based approaches to build a WordNet like resource automatically (Hearst, 1992; Caraballo, 1999; Berland and Charniak, 1999). Also, several researchers have applied it to resolving different types of bridging anaphora (Clark, 1975). Poesio *et al.* (2002) have proposed extracting lexical knowledge about part-of relations using Hearst-style patterns and applied it to the task of resolving bridging references. Poesio *et al.* (2004) have suggested using Google as a source of computing lexical distance between antecedent and definite NP for mereological bridging references (references referring to parts of an object already introduced). Markert *et al.* (2003) have applied relations extracted from lexico-syntactic patterns such as *'X and other Ys'* for Other-Anaphora (referential NPs with modifiers *other* or *another*) and for bridging involving meronymy.

There has generally been a lack of work in the existing literature for automatically building lexical resources for definite anaphora resolution involving hyponyms relations such as presented in Example (1). However, this issue was recently addressed by Markert and Nissim (2005) by extending their work on Other-Anaphora using lexico syntactic pattern *'X and other Y's* to antecedent selection for definite NP coreference. However, our task is more challenging since the anaphoric definite NPs in our test set include only hypernym anaphors without including the much simpler cases of headword repetition and other instances of string matching. For direct evaluation, we also implemented their corpus-based approach and compared it with our models on identical test data.

We also describe and evaluate a mechanism for combining the knowledge obtained from WordNet and the six corpus-based approaches investigated here. The resulting models are able to overcome the weaknesses of a WordNet-only model and substantially outperforms any of the individual models.

3 Models for Lexical Acquisition

3.1 TheY-Model

Our algorithm is motivated by the observation that in a discourse, the use of the definite article ("the") in a non-deictic context is primarily licensed if the concept has already been mentioned in the text. Hence a sentence such as "The drug is very expensive" generally implies that either the word *drug* itself was previously mentioned (e.g. "He is taking a new drug for his high cholesterol.") or a hyponym of *drug* was

[1] The test examples were selected as follows: First, all the sentences containing definite NP "*The Y*" were extracted from the corpus. Then, the sentences containing instances of anaphoric definite NPs were kept and other cases of definite expressions (like existential NPs "*The White House*","*The weather*") were discarded. From this anaphoric set of sentences, 177 sentence instances covering 13 distinct hypernyms were randomly selected as the test set and annotated for the correct antecedent by human judges.

previously mentioned (e.g. "He is taking Lipitor for his high cholesterol."). Because it is straightforward to filter out the former case by string matching, the residual instances of the phrase "the drug" (without previous mentions of the word "drug" in the discourse) are likely to be instances of hypernymic definite anaphora. We can then determine which nouns earlier in the discourse (e.g. *Lipitor*) are likely antecedents by unsupervised statistical co-occurrence modeling aggregated over the entire corpus. All we need is a large corpus without any anaphora annotation and a basic tool for noun tagging and NP head annotation. The detailed algorithm is as follows:

1. Find each sentence in the training corpus that contains a definite NP (*'the Y'*) and does not contain *'a Y'*, *'an Y'* or other instantiations of *Y*[2] appearing before the definite NP within a fixed window.[3]
2. In the sentences that pass the above definite NP and *a/an test*, regard all the head words (X) occurring in the current sentence before the definite NP and the ones occurring in previous two sentences as potential antecedents.
3. Count the frequency c(X,Y) for each pair obtained in the above two steps and pre-store it in a table.[4] The frequency table can be modified to give other scores for pair(X,Y) such as standard TF-IDF and Mutual Information scores.
4. Given a test sentence having an anaphoric definite NP Y, consider the nouns appearing before Y within a fixed window as potential antecedents. Rank the candidates by their precomputed co-occurence measures as computed in Step 3.

Since we consider *all* head words preceding the definite NP as potential correct antecedents, the raw frequency of the pair (X,Y) can be very noisy. This can be seen clearly in Table 1, where the first column shows the top potential antecedents of definite NP *the drug* as given by raw frequency. We normalize the raw frequency using standard TF-IDF and Mutual Information scores to filter the noisy pairs.[5] In Table 2, we report our results for antecedent selection using Raw frequency c(X,Y), TF-IDF [6] and MI in isolation. *Accuracy* is the fraction of total examples that were assigned the correct antecedent and $Accuracy_{tag}$ is the same excluding the examples that had POS tagging errors for the correct antecedent.[7] *Av Rank* is the rank of the true antecedent averaged over the number of test examples.[8] Based on the above experiment, the rest of this paper assumes Mutual Information scoring technique for TheY-Model.

Rank	Raw freq	TF-IDF	**MI**
1	today	kilogram	**amphetamine**
2	police	heroin	**cannabis**
3	kilogram	police	**cocaine**
4	year	cocaine	**heroin**
5	heroin	today	**marijuana**
6	dollar	trafficker	**pill**
7	country	officer	**hashish**
8	official	amphetamine	**tablet**

Table 1: A sample of ranked hyponyms proposed for the definite NP ***The drug*** by TheY-Model illustrating the differences in weighting methods.

	Acc	Acc_{tag}	Av Rank
MI	**0.531**	**0.577**	**4.82**
TF-IDF	0.175	0.190	6.63
Raw Freq	0.113	0.123	7.61

Table 2: Results using different normalization techniques for the TheY-Model in isolation. (60 million word corpus)

[2]While matching for both *'the Y'* and *'a/an Y'*, we also account for Nouns getting modified by other words such as adjectives. Thus *'the Y'* will still match to *'the green and big Y'*.

[3]Window size was set to two sentences, we also experimented with a larger window size of five sentences and the results obtained were similar.

[4]Note that the count c(X,Y) is asymmetric

[5]Note that $MI(X,Y) = log\ \frac{P(X,Y)}{P(X)P(Y)}$ and this is directly proportional to $P(Y|X) = \frac{c(X,Y)}{c(X)}$ for a fixed Y. Thus, we can simply use this conditional probability during implementation since the definite NP Y is fixed for the task of antecedent selection.

[6]For the purposes of TF-IDF computation, document frequency df(X) is defined as the number of unique definite NPs for which X appears as an antecedent.

[7]Since the POS tagging was done automatically, it is possible for any model to miss the correct antecedent because it was not tagged correctly as a noun in the first place. There were 14 such examples in the test set and none of the model variants can find the correct antecdent in these instances.

[8]Knowing average rank can be useful when a n-best ranked list from coreference task is used as an input to other downstream tasks such as information extraction.

	Acc	Acc_{tag}	Av Rank
TheY+WN	**0.695**	**0.755**	3.37
WordNet	0.593	0.644	**3.29**
TheY	0.531	0.577	4.82

Table 3: Accuracy and Average Rank showing combined model performance on the antecedent selection task. Corpus Size: 60 million words.

3.2 WordNet-Model (WN)

Because WordNet is considered as a standard resource of lexical knowledge and is often used in coreference tasks, it is useful to know how well corpus-based approaches perform as compared to a standard model based on the WordNet (version 2.0).[9] The algorithm for the WordNet-Model is as follows:

Given a definite NP Y and its potential antecedent X, choose X if it occurs as a hyponym (either direct or indirect inheritance) of Y. If multiple potential antecedents occur in the hierarchy of Y, choose the one that is closest in the hierarchy.

3.3 Combination: TheY+WordNet Model

Most of the literature on using lexical resources for definite anaphora has focused on using individual models (either corpus-based or manually build resources such as WordNet) for antecedent selection. Some of the difficulties with using WordNet is its limited coverage and its lack of empirical ranking model. We propose a combination of TheY-Model and WordNet-Model to overcome these problems. Essentially, we rerank the hypotheses found in WordNet-Model based on ranks of TheY-model or use a backoff scheme if WordNet-Model does not return an answer due to its limited coverage. Given a definite NP Y and a set of potential antecedents Xs the detailed algorithm is specified as follows:

1. *Rerank with TheY-Model:* Rerank the potential antecedents found in the WordNet-Model table by assiging them the ranks given by TheY-Model. If TheY-Model does not return a rank for a potential antecedent, use the rank given by the WordNet-Model. Now pick the top ranked antecedent after reranking.

2. *Backoff:* If none of the potential antecedents were found in the WordNet-Model then pick the correct antecedent from the ranked list of The-Y model. If none of the models return an answer then assign ranks uniformly at random.

The above algorithm harnesses the strength of WordNet-Model to identify good hyponyms and the strength of TheY-model to identify which are more likely to be used as an antecedent. Note that this combination algorithm can be applied using any corpus-based technique to account for poor-ranking and low-coverage problems of WordNet and the Sections 3.4, 3.5 and 3.6 will show the results for backing off to a Hearst-style hypernym model. Table 4 shows the decisions made by TheY-model, WordNet-Model and the combined model for a sample of test examples. It is interesting to see how both the models mutually complement each other in these decisions. Table 3 shows the results for the models presented so far using a 60 million word training text from the Gigaword corpus. The combined model results in a substantially better accuracy than the individual WordNet-Model and TheY-Model, indicating its strong merit for the antecedent selection task.[10]

3.4 OtherY-Model$_{freq}$

This model is a reimplementation of the corpus-based algorithm proposed by Markert and Nissim (2005) for the equivalent task of antecedent selection for definite NP coreference. We implement their approach of using the lexico-syntactic pattern *X and A* other B* Y*$\{pl\}$ for extracting (X,Y) pairs.The *A** and *B** allow for adjectives or other modifiers to be placed in between the pattern. The model presented in their article uses the raw frequency as the criteria for selecting the antecedent.

3.5 OtherY-Model$_{MI}$(normalized)

We normalize the OtherY-Model using Mutual Information scoring method. Although Markert and Nissim (2005) report that using Mutual Information performs similar to using raw frequency, Table 5 shows that using Mutual Information makes a substantial impact on results using large training corpora relative to using raw frequency.

[9] We also computed the accuracy using a weaker baseline, namely, selecting the closest previous headword as the correct antecedent. This recency based baseline obtained a low accuracy of 15% and hence we used the stronger WordNet based model for comparison purposes.

[10] The claim is statistically significant with a $p < 0.01$ obtained by sign-test

Summary	Keyword (Def. Ana)	True Antecedent	TheY Choice	Truth Rank	WordNet Choice	Truth Rank	TheY+WN Choice	Truth Rank
Both correct	metal	gold	gold	1	gold	1	gold	1
	sport	soccer	soccer	1	soccer	1	soccer	1
TheY-Model helps	drug	steroid	steroid	1	NA	NA	steroid	1
	drug	azt	azt	1	medication	2	azt	1
WN-Model helps	instrument	trumpet	king	10	trumpet	1	trumpet	1
	drug	naltrexone	alcohol	14	naltrexone	1	naltrexone	1
Both incorrect	weapon	bomb	artillery	3	NA	NA	artillery	3
	instrument	voice	music	9	NA	NA	music	9

Table 4: A sample of output from different models on antecedent selection (60 million word corpus).

3.6 Combination: TheY+OtherY$_{MI}$ Model

Our two corpus-based approaches (TheY and OtherY) make use of different linguistic phenomena and it would be interesting to see whether they are complementary in nature. We used a similar combination algorithm as in Section 3.3 with the WordNet-Model replaced with the OtherY-Model for hypernym filtering, and we used the noisy TheY-Model for reranking and backoff. The results for this approach are showed as the entry TheY+OtherY$_{MI}$ in Table 5. We also implemented a combination (OtherY+WN) of Other-Y model and WordNet-Model by replacing TheY-Model with OtherY-Model in the algorithm described in Section 3.3. The respective results are indicated as OtherY+WN entry in Table 5.

4 Further Anaphora Resolution Results

Table 5 summarizes results obtained from all the models defined in Section 3 on three different sizes of training unlabeled corpora (from Gigaword corpus). The models are listed from high accuracy to low accuracy order. The OtherY-Model performs particularly poorly on smaller data sizes, where coverage of the Hearst-style patterns maybe limited, as also observed by Berland and Charniak (1999). We further find that the Markert and Nissim (2005) OtherY-Model and our MI-based improvement do show substantial relative performance growth at increased corpus sizes, although they still underperform our basic TheY-Model at all tested corpus sizes. Also, the combination of corpus-based models (TheY-Model+OtherY-model) does indeed performs better than either of them in isolation. Finally, note that the basic TheY-algorithm still does

	Acc	Acc$_{tag}$	Av Rank
60 million words			
TheY+WN	**0.695**	**0.755**	3.37
OtherY$_{MI}$+WN	0.633	0.687	**3.04**
WordNet	0.593	0.644	3.29
TheY	0.531	0.577	4.82
TheY+OtherY$_{MI}$	0.497	0.540	4.96
OtherY$_{MI}$	0.356	0.387	5.38
OtherY$_{freq}$	0.350	0.380	5.39
230 million words			
TheY+WN	**0.678**	**0.736**	3.61
OtherY$_{MI}$+WN	0.650	0.705	**2.99**
WordNet	0.593	0.644	3.29
TheY+OtherY$_{MI}$	0.559	0.607	4.50
TheY	0.519	0.564	4.64
OtherY$_{MI}$	0.503	0.546	4.37
OtherY$_{freq}$	0.418	0.454	4.52
380 million words			
TheY+WN	**0.695**	**0.755**	3.47
OtherY$_{MI}$+WN	0.644	0.699	**3.03**
WordNet	0.593	0.644	3.29
TheY+OtherY$_{MI}$	0.554	0.601	4.20
TheY	0.537	0.583	4.26
OtherY$_{MI}$	0.525	0.571	4.20
OtherY$_{freq}$	0.446	0.485	4.36

Table 5: Accuracy and Average Rank of Models defined in Section 3 on the antecedent selection task.

relatively well by itself on smaller corpus sizes, suggesting its merit on resource-limited languages with smaller available online text collections and the unavailability of WordNet. The combined models of WordNet-Model with the two corpus-based approaches still significantly ($p < 0.01$) outperform any of the other individual models.[11]

5 Generation Task

Having shown positive results for the task of antecedent selection, we turn to a more difficult task, namely *generating* an anaphoric definite NP given a nominal antecedent. In Example (1), this would correspond to generating *"the drug"* as an anaphor knowing that the antecedent is *pseudoephedrine*. This task clearly has many applications: current generation systems often limit their anaphoric usage to pronouns and thus an automatic system that does well on hypernymic definite NP generation can directly be helpful. It also has strong potential application in abstractive summarization where rewriting a fluent passage requires a good model of anaphoric usage.

There are many interesting challenges in this problem: first of all, there maybe be multiple acceptable choices for definite anaphor given a particular antecedent, complicating automatic evaluation. Second, when a system generates a definite anaphora, the space of potential candidates is essentially unbounded, unlike in antecdent selection, where it is limited only to the number of potential antecedents in prior context. In spite of the complex nature of this problem, our experiments with the human judgements, WordNet and corpus-based approaches show a simple feasible solution. We evaluate our automatic approaches based on exact-match agreement with definite anaphora actually used in the corpus (accuracy) and also by agreement with definite anaphora predicted independently by a human judge in an absence of context.

[11]Note that syntactic co-reference candidate filters such as the Hobbs algorithm were not utilized in this study. To assess the performance implications, the Hobbs algorithm was applied to a randomly selected 100-instance subset of the test data. Although the Hobbs algorithm frequently pruned at least one of the coreference candidates, in only 2% of the data did such candidate filtering change system output. However, since both of these changes were improvements, it could be worthwhile to utilize Hobbs filtering in future work, although the gains would likely be modest.

5.1 Human experiment

We extracted a total of 103 <true antecedent, definite NP> pairs from the set of test instances used in the resolution task. Then we asked a human judge (a native speaker of English) to predict a parent class of the antecedent that could act as a good definite anaphora choice in general, independent of a particular context. Thus, the actual corpus sentence containing the antecedent and definite NP and its context was not provided to the judge. We took the predictions provided by the judge and matched them with the actual definite NPs used in the corpus. The agreement between corpus and the human judge was 79% which can thus be considered as an upper bound of algorithm performance. Table 7 shows a sample of decisions made by the human and how they agree with the definite NPs observed in the corpus. It is interesting to note the challenge of the sense variation and figurative usage. For example, *"corruption"* is refered to as a *"tool"* in the actual corpus anaphora, a metaphoric usage that would be difficult to predict unless given the usage sentence and its context. However, a human agreement of 79% indicate that such instances are relatively rare and the task of predicting a definite anaphor without its context is viable. In general, it appears from our experiements that humans tend to select from a relatively small set of parent classes when generating hypernymic definite anaphora. Furthermore, there appears to be a relatively context-independent concept of the "natural" level in the hypernym hierarchy for generating anaphors. For example, although <*"alkaloid", "organic compound", "compound", "substance", "entity"*> are all hypernyms of *"Pseudoephederine"* in WordNet, *"the drug"* appears to be the preferred hypernym for definite anaphora in the data, with the other alternatives being either too specific or too general to be natural. This natural level appears to be difficult to define by rule. For example, using just the immediate parent hypernym in the WordNet hierarchy only achieves 4% match with the corpus data for definite anaphor generation.

5.2 Algorithms

The following sections presents our corpus-based algorithms as more effective alternatives.

	Agreement w/ human judge	Agreement w/ corpus
TheY+OtherY+WN	**47%**	**46%**
OtherY +WN	43%	43%
TheY+WN	42%	37%
TheY +OtherY	39%	36%
OtherY	39%	36%
WordNet	4%	4%
Human judge	100%	79%
Corpus	79%	100%

Table 6: Agreement of different generation models with human judge and with definite NP used in the corpus.

Antecedent	Corpus Def Ana	Human Choice	TheY+OtherY +WN
racing	sport	sport	sport
azt	drug	drug	drug
missile	weapon	weapon	weapon
alligator	animal	animal	animal
steel	metal	metal	metal
osteporosis	disease	disease	condition
grenade	device	weapon	device
baikonur	site	city	station
corruption	tool	crime	activity

Table 7: Sample of decisions made by human judge and our best performing model (TheY+OtherY+WN) on the generation task.

5.2.1 Individual Models

For the corpus-based approaches, the TheY-Model and OtherY-Model were trained in the same manner as for the antecedent selection task. The only difference was that in the generation case, the frequency statistics were reversed to provide a hypernym given a hyponym. Additionally, we found that raw frequency outperformed either TF-IDF or Mutual Information and was used for all results in Table 6.
The stand-alone WordNet model is also very simple: Given an antecedent, we lookup its direct hypernym (using first sense) in the WordNet and use it as the definite NP, for lack of a better rule for preferred hypernym location.

5.2.2 Combining corpus-based approaches and WordNet

Each of the corpus-based approaches was combined with WordNet resulting in two different models as follows: Given an antecedent X, the corpus-based approach looks up in its table the hypernym of X, for example Y, and only produces Y as the output if Y also occurs in the WordNet as hypernym. Thus WordNet is used as a filtering tool for detecting viable hypernyms. This combination resulted in two models: *'TheY+WN'* and *'OtherY+WN'*.
We also combined all the three approaches, *'TheY'*, *'OtherY'* and WordNet resulting in a single model *'TheY+OtherY+WN'*. This was done as follows: We first combine the models *'TheY'* and *'OtherY'* using a backoff model. The first priority is to use the hypernym from the model *'OtherY'*, if not found then use the hypernym from the model *'TheY'*. Given a definite NP from the backoff model, apply the WordNet filtering technique, specifically, choose it as the correct definite NP if it also occurs as a hypernym in the WordNet hierarchy of the antecedent.

5.3 Evaluation of Anaphor Generation

We evaluated the resulting algorithms from Section 5.2 on the definite NP prediction task as described earlier. Table 6 shows the agreement of the algorithm predictions with the human judge as well as with the definite NP actually observed in the corpus. It is interesting to see that WordNet by itself performs very poorly on this task since it does not have any word-specific mechanism to choose the correct level in the hierarchy and the correct word sense for selecting the hypernym. However, when combined with our corpus-based approaches, the agreement increases substantially indicating that the corpus-based approaches are effectively filtering the space of hypernyms that can be used as natural classes. Likewise, WordNet helps to filter the noisy hypernyms from the corpus predictions. Thus, this interplay between the corpus-based and WordNet algorithm works out nicely, resulting in the best model being a combination of all three individual models and achieving a substantially better agreement with both the corpus and human judge than any of the individual models. Table 7 shows decisions made by this algorithm on a sample test data.

6 Conclusion

This paper provides a successful solution to the problem of incomplete lexical resources for definite anaphora resolution and further demonstrates how the resources built for resolution can be naturally extended for the less studied task of anaphora generation. We first presented a simple and noisy corpus-based approach based on globally modeling head-word co-occurrence around likely anaphoric definite NPs. This was shown to outperform a recent approach by Markert and Nissim (2005) that makes use of standard Hearst-style patterns extracting hypernyms for the same task. Even with a relatively small training corpora, our simple TheY-model was able to achieve relatively high accuracy, making it suitable for resource-limited languages where annotated training corpora and full WordNets are likely not available. We then evaluated several variants of this algorithm based on model combination techniques. The best combined model was shown to exceed 75% accuracy on the resolution task, beating any of the individual models. On the much harder anaphora generation task, where the stand-alone WordNet-based model only achieved an accuracy of 4%, we showed that our algorithms can achieve 35%-47% accuracy on blind exact-match evaluation, thus motivating the use of such corpus-based learning approaches on the generation task as well.

Acknowledgements

Thanks to Charles Schafer for sharing his tools on POS/Headword tagging for the Gigaword corpus.

References

M. Berland and E. Charniak. 1999. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 57–64.

S. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 120–126.

H. H. Clark. 1975. Bridging. In *Proceedings of the Conference on Theoretical Issues in Natural Language Processing*, pages 169–174.

D. Connoly, J. D. Burger, and D. S. Day. 1997. A machine learning approach to anaphoric reference. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 133–144.

D. Graff, J. Kong, K. Chen, and K. Maeda. 2005. English Gigaword Second Edition. Linguistic Data Consortium, catalog number LDC2005T12.

S. Harabagiu, R. Bunescu, and S. J. Maiorano. 2001. Text and knowledge mining for coreference resolution. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 55–62.

M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545.

J. Hobbs. 1978. Resolving pronoun references. *Lingua*, 44:311–338.

K. Markert and M. Nissim. 2005. Comparing knowledge sources for nominal anaphora resolution. *Computational Linguistics*, 31(3):367–402.

K. Markert, M. Nissim, and N. N. Modjeska. 2003. Using the web for nominal anaphora resolution. In *Proceedings of the EACL Workshop on the Computational Treatment of Anaphora*, pages 39–46.

J. Meyer and R. Dale. 2002. Mining a corpus to support associative anaphora resolution. In *Proceedings of the Fourth International Conference on Discourse Anaphora and Anaphor Resolution*.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

M. Poesio, R. Vieira, and S. Teufel. 1997. Resolving bridging references in unrestricted text. In *Proceedings of the ACL Workshop on Operational Factors in Robust Anaphora*, pages 1–6.

M. Poesio, T. Ishikawa, S. Schulte im Walde, and R. Viera. 2002. Acquiring lexical knowledge for anaphora resolution. In *Proccedings of the Third Conference on Language Resources and Evaluation*, pages 1220–1224.

M. Poesio, R. Mehta, A. Maroudas, and J. Hitzeman. 2004. Learning to resolve bridging references. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 143–150.

W. M. Soon, H. T. Ng, and D. C. Y. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

M. Strube, S. Rapp, and C. Müller. 2002. The influence of minimum edit distance on reference resolution. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 312–319.

R. Vieira and M. Poesio. 2000. An empirically-based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–593.

X. Yang, G. Zhou, J. Su, and C. L. Tan. 2003. Coreference resolution using competition learning approach. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 176–183.

Investigating Lexical Substitution Scoring for Subtitle Generation

Oren Glickman and **Ido Dagan**
Computer Science Department
Bar Ilan University
Ramat Gan, Israel
{glikmao,dagan}@cs.biu.ac.il

Mikaela Keller and **Samy Bengio**
IDIAP Research Institute
Martigny,
Switzerland
{mkeller,bengio}@idiap.ch

Walter Daelemans
CNTS
Antwerp, Belgium
walter.daelemans@ua.ac.be

Abstract

This paper investigates an isolated setting of the lexical substitution task of replacing words with their synonyms. In particular, we examine this problem in the setting of subtitle generation and evaluate state of the art scoring methods that predict the validity of a given substitution. The paper evaluates two context independent models and two contextual models. The major findings suggest that distributional similarity provides a useful complementary estimate for the likelihood that two Wordnet synonyms are indeed substitutable, while proper modeling of contextual constraints is still a challenging task for future research.

1 Introduction

Lexical substitution - the task of replacing a word with another one that conveys the same meaning - is a prominent task in many Natural Language Processing (NLP) applications. For example, in query expansion for information retrieval a query is augmented with synonyms of the original query words, aiming to retrieve documents that contain these synonyms (Voorhees, 1994). Similarly, lexical substitutions are applied in question answering to identify answer passages that express the sought answer in different terms than the original question. In natural language generation it is common to seek lexical alternatives for the same meaning in order to reduce lexical repetitions. In general, lexical substitution aims to preserve a desired meaning while coping with the lexical variability of expressing that meaning. Lexical substitution can thus be viewed within the general framework of recognizing entailment between text segments (Dagan et al., 2005), as modeling entailment relations at the lexical level.

In this paper we examine the lexical substitution problem within a specific setting of text compression for subtitle generation (Daelemans et al., 2004). Subtitle generation is the task of generating target language TV subtitles for video recordings of a source language speech. The subtitles should be of restricted length, which is often shorter than the full translation of the original speech, yet they should maintain as much as possible the meaning of the original content. In a typical (automated) subtitling process the original speech is first translated fully into the target language and then the target translation is compressed to optimize the length requirements. One of the techniques employed in the text compression phase is to replace a target language word in the original translation with a shorter synonym of it, thus reducing the character length of the subtitle. This is a typical lexical substitution task, which resembles similar operations in other text compression and generation tasks (e.g. (Knight and Marcu, 2002)).

This paper investigates the task of assigning likelihood scores for the correctness of such lexical substitutions, in which words in the original translation are replaced with shorter synonyms. In our experiments we use WordNet as a source of candidate synonyms for substitution. The goal is to score the likelihood that the substitution is admissible, i.e. yielding a valid sentence that preserves the original meaning. The focus of this paper is thus to utilize the subtitling setting in order to investigate lexical sub-

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 45–52, New York City, June 2006.

stitution models in isolation, unlike most previous literature in which this sub-task has been embedded in larger systems and was not evaluated directly.

We examine four statistical scoring models, of two types. Context independent models score the general likelihood that the original word is "replaceable" with the candidate synonym, in an arbitrary context. That is, trying to filter relatively bizarre synonyms, often of rare senses, which are abundant in WordNet but are unlikely to yield valid substitutions. Contextual models score the "fitness" of the replacing word within the context of the sentence, in order to filter out synonyms of senses of the original word that are not the right sense in the given context.

We set up an experiment using actual subtitling data and human judgements and evaluate the different scoring methods. Our findings suggest the dominance, in this setting, of generic context-independent scoring. In particular, considering distributional similarity amongst WordNet synonyms seems effective for identifying candidate substitutions that are indeed likely to be applicable in actual texts. Thus, while distributional similarity alone is known to be too noisy as a sole basis for meaning-preserving substitutions, its combination with WordNet allows reducing the noise caused by the many WordNet synonyms that are unlikely to correspond to valid substitutions.

2 Background and Setting

2.1 Subtitling

Automatic generation of subtitles is a summarization task at the level of individual sentences or occasionally of a few contiguous sentences. Limitations on reading speed of viewers and on the size of the screen that can be filled with text without the image becoming too cluttered, are the constraints that dynamically determine the amount of compression in characters that should be achieved in transforming the transcript into subtitles. Subtitling is not a trivial task, and is expensive and time-consuming when experts have to carry it out manually. As for other NLP tasks, both statistical (machine learning) and linguistic knowledge-based techniques have been considered for this problem. Examples of the former are (Knight and Marcu, 2002; Hori et al., 2002), and of the latter are (Grefenstette, 1998; Jing and McKeown, 1999). A comparison of both approaches in the context of a Dutch subtitling system is provided in (Daelemans et al., 2004). The required sentence simplification is achieved either by deleting material, or by paraphrasing parts of the sentence into shorter expressions with the same meaning. As a special case of the latter, lexical substitution is often used to achieve a compression target by substituting a word by a shorter synonym. It is on this subtask that we focus in this paper. Table 1 provides a few examples. E.g. by substituting "happen" by "occur" (example 3), one character is saved without affecting the sentence meaning .

2.2 Experimental Setting

The data used in our experiments was collected in the context of the MUSA (Multilingual Subtitling of Multimedia Content) project (Piperidis et al., 2004)[1] and was kindly provided for the current study. The data was provided by the BBC in the form of *Horizon* documentary transcripts with the corresponding audio and video. The data for two documentaries was used to create a dataset consisting of sentences from the transcripts and the corresponding substitution examples in which selected words are substituted by a shorter Wordnet synonym. More concretely, a *substitution example* thus consists of an original sentence $s = w_1 \dots w_i \dots w_n$, a specific *source* word w_i in the sentence and a *target* (shorter) WordNet synonym w' to substitute the source. See Table 1 for examples. The dataset consists of 918 substitution examples originating from 231 different sentences.

An annotation environment was developed to allow efficient annotation of the substitution examples with the classes *true* (admissible substitution, in the given context) or *false* (inadmissible substitution). About 40% of the examples were judged as true. Part of the data was annotated by an additional annotator to compute annotator agreement. The Kappa score turned out to be 0.65, corresponding to "Substantial Agreement" (Landis and Koch, 1997). Since some of the methods we are comparing need tuning we held out a random subset of 31 original sentences (with 121 corresponding examples) for development and kept for testing the resulting 797 substitution ex-

[1] http://sinfos.ilsp.gr/musa/

id	sentence	source	target	judgment
1	The answer may be found in the behaviour of animals.	answer	reply	false
2	. . . and the answer to that was - Yes	answer	reply	true
3	We then wanted to know what would happen if	happen	occur	true
4	we delay the movement of the subject's left hand	subject	topic	false
5		subject	theme	false
6	people weren't laughing they were going stone sober.	stone	rock	false
7	if we can identify a place where the seizures are coming from then we can go in and remove just that small area.	identify	place	false
8	my approach has been the first to look at the actual structure of the laugh sound.	approach	attack	false
9	He quickly ran into an unexpected problem.	problem	job	false
10	today American children consume 5 times more Ritalin than the rest of the world combined	consume	devour	false

Table 1: Substitution examples from the dataset along with their annotations

amples from the remaining 200 sentences.

3 Compared Scoring Models

We compare methods for scoring lexical substitutions. These methods assign a score which is expected to correspond to the likelihood that the synonym substitution results in a valid subtitle which preserves the main meaning of the original sentence.

We examine four statistical scoring models, of two types. The context independent models score the general likelihood that the source word can be replaced with the target synonym regardless of the context in which the word appears. Contextual models, on the other hand, score the fitness of the target word within the given context.

3.1 Context Independent Models

Even though synonyms are substitutable in theory, in practice there are many rare synonyms for which the likelihood of substitution is very low and will be substitutable only in obscure contexts. For example, although there are contexts in which the word *job* is a synonym of the word *problem*[2], this is not typically the case and overall job is not a good target substitution for the source *problem* (see example 9 in Table 1). For this reason synonym thesauruses such as WordNet tend to be rather noisy for practical purposes, raising the need to score such synonym substitutions and accordingly prioritize substitutions that are more likely to be valid in an arbitrary context.

As representative approaches for addressing this problem, we chose two methods that rely on statistical information of two types: supervised sense distributions from SemCor and unsupervised distributional similarity.

3.1.1 WordNet based Sense Frequencies (semcor)

The obvious reason that a target synonym cannot substitute a source in some context is if the source appears in a different sense than the one in which it is synonymous with the target. This means that a priori, synonyms of frequent senses of a source word are more likely to provide correct substitutions than synonyms of the word's infrequent senses.

To estimate such likelihood, our first measure is based on sense frequencies from SemCor (Miller et al., 1993), a corpus annotated with Wordnet senses. For a given source word u and target synonym v the score is calculated as the percentage of occurrences of u in SemCor for which the annotated synset contains v (i.e. u's occurrences in which its sense is synonymous with v). This corresponds to the prior probability estimate that an occurrence of u (in an arbitrary context) is actually a synonym of v. Therefore it is suitable as a prior score for lexical substitution.[3]

3.1.2 Distributional Similarity (sim)

The SemCor based method relies on a supervised approach and requires a sense annotated corpus. Our

[2]WordNet lists job as a possible member of the synset for a state of difficulty that needs to be resolved, as might be used in sentences like "it is always a job to contact him"

[3]Note that WordNet semantic distance measures such as those compared in (Budanitsky and Hirst, 2001) are not applicable here since they measure similarity between synsets rather than between synonymous words within a single synset.

second method uses an unsupervised distributional similarity measure to score synonym substitutions. Such measures are based on the general idea of Harris' Distributional Hypothesis, suggesting that words that occur within similar contexts are semantically similar (Harris, 1968).

As a representative of this approach we use Lin's dependency-based distributional similarity database. Lin's database was created using the particular distributional similarity measure in (Lin, 1998), applied to a large corpus of news data (64 million words) [4]. Two words obtain a high similarity score if they occur often in the same contexts, as captured by syntactic dependency relations. For example, two verbs will be considered similar if they have large common sets of modifying subjects, objects, adverbs etc.

Distributional similarity does not capture directly meaning equivalence and entailment but rather a looser notion of meaning similarity (Geffet and Dagan, 2005). It is typical that non substitutable words such as antonyms or co-hyponyms obtain high similarity scores. However, in our setting we apply the similarity score only for WordNet synonyms in which it is known a priori that they are substitutable is some contexts. Distributional similarity may thus capture the statistical degree to which the two words are substitutable in practice. In fact, it has been shown that prominence in similarity score corresponds to sense frequency, which was suggested as the basis for an unsupervised method for identifying the most frequent sense of a word (McCarthy et al., 2004).

3.2 Contextual Models

Contextual models score lexical substitutions based on the context of the sentence. Such models try to estimate the likelihood that the target word could potentially occur in the given context of the source word and thus may replace it. More concretely, for a given substitution example consisting of an original sentence $s = w_1 \dots w_i \dots w_n$, and a designated source word w_i, the contextual models we consider assign a score to the substitution based solely on the target synonym v and the context of the source word in the original sentence, $\{w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_n\}$, which is represented in a bag-of-words format.

Apparently, this setting was not investigated much in the context of lexical substitution in the NLP literature. We chose to evaluate two recently proposed models that address exactly the task at hand: the first model was proposed in the context of lexical modeling of textual entailment, using a generative Naïve Bayes approach; the second model was proposed in the context of machine learning for information retrieval, using a discriminative neural network approach. The two models were trained on the (unannotated) sentences of the BNC 100 million word corpus (Burnard, 1995) in bag-of-words format. The corpus was broken into sentences, tokenized, lemmatized and stop words and tokens appearing only once were removed. While training of these models is done in an unsupervised manner, using unlabeled data, some parameter tuning was performed using the small development set described in Section 2.

3.2.1 Bayesian Model (bayes)

The first contextual model we examine is the one proposed in (Glickman et al., 2005) to model textual entailment at the lexical level. For a given target word this unsupervised model takes a binary text categorization approach. Each vocabulary word is considered a class, and contexts are classified as to whether the given target word is likely to occur in them. Taking a probabilistic Naïve-Bayes approach the model estimates the conditional probability of the target word given the context based on corpus co-occurrence statistics. We adapted and implemented this algorithm and trained the model on the sentences of the BNC corpus.

For a bag-of-words context $C = \{w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_n\}$ and target word v the Naïve Bayes probability estimation for the conditional probability of a word v may occur in a given a context C is as follows:

$$\mathrm{P}(v|C) = \frac{\mathrm{P}(C|v)\,\mathrm{P}(v)}{\mathrm{P}(C|v)\,\mathrm{P}(v)+\mathrm{P}(C|\neg v)\,\mathrm{P}(\neg v)} \approx \frac{\mathrm{P}(v)\prod_{w\in C}\mathrm{P}(w|v)}{\mathrm{P}(v)\prod_{w\in C}\mathrm{P}(w|v)+\mathrm{P}(\neg v)\prod_{w\in C}\mathrm{P}(w|\neg v)} \quad (1)$$

where $\mathrm{P}(w|v)$ is the probability that a word w appears in the context of a sentence containing v and correspondingly $\mathrm{P}(w|\neg v)$ is the probability that w

[4] available at http://www.cs.ualberta.ca/~lindek/downloads.htm

appears in a sentence not containing v. The probability estimates were obtained from the processed BNC corpus as follows:

$$\mathrm{P}(w|v) = \frac{|w \text{ appears in sentences containing } v|}{|\text{words in sentences containing } v|}$$
$$\mathrm{P}(w|\neg v) = \frac{|w \text{ occurs in sentences not containing } v|}{|\text{words in sentences not containing } v|}$$

To avoid 0 probabilities these estimates were smoothed by adding a small constant to all counts and normalizing accordingly. The constant value was tuned using the development set to maximize average precision (see Section 4.1). The estimated probability, $\mathrm{P}(v|C)$, was used as the confidence score for each substitution example.

3.2.2 Neural Network Model (nntr)

As a second contextual model we evaluated the Neural Network for Text Representation (NNTR) proposed in (Keller and Bengio, 2005). NNTR is a discriminative approach which aims at modeling how likely a given word v is in the context of a piece of text C, while learning a more compact representation of reduced dimensionality for both v and C.

NNTR is composed of 3 Multilayer Perceptrons, noted $mlp_{\mathcal{A}}()$, $mlp_{\mathcal{B}}()$ and $mlp_{\mathcal{C}}()$, connected as follow:

$$NNTR(v, C) = mlp_{\mathcal{C}}[mlp_{\mathcal{A}}(v), mlp_{\mathcal{B}}(C)].$$

$mlp_{\mathcal{A}}(v)$ and $mlp_{\mathcal{B}}(C)$ project respectively the vector space representation of the word and text into a more compact space of lower dimensionality. $mlp_{\mathcal{C}}()$ takes as input the new representations of v and C and outputs a score for the contextual relevance of v to C.

As training data, couples (v,C) from the BNC corpus are provided to the learning scheme. The target training value for the output of the system is 1 if v is indeed in C and -1 otherwise. The hope is that the neural network will be able to generalize to words which are not in the piece of text but are likely to be related to it.

In essence, this model is trained by minimizing the weighted sum of the hinge loss function over negative and positive couples, using stochastic Gradient Descent (see (Keller and Bengio, 2005) for further details). The small held out development set of the substitution dataset was used to tune the hyperparameters of the model, maximizing average precision (see Section 4.1). For simplicity $mlp_{\mathcal{A}}()$ and $mlp_{\mathcal{B}}()$ were reduced to Perceptrons. The output size of $mlp_{\mathcal{A}}()$ was set to 20, $mlp_{\mathcal{B}}()$ to 100 and the number of hidden units of $mlp_{\mathcal{C}}()$ was set to 500.

There are a couple of important conceptual differences of the discriminative NNTR model compared to the generative Bayesian model described above. First, the relevancy of v to C in NNTR is inferred in a more compact representation space of reduced dimensionality, which may enable a higher degree of generalization. Second, in NNTR we are able to control the capacity of the model in terms of number of parameters, enabling better control to achieve an optimal generalization level with respect to the training data (avoiding over or under fitting).

4 Empirical Results

4.1 Evaluation Measures

We compare the lexical substitution scoring methods using two evaluation measures, offering two different perspectives of evaluation.

4.1.1 Accuracy

The first evaluation measure is motivated by simulating a decision step of a subtitling system, in which the best scoring lexical substitution is selected for each given sentence. Such decision may correspond to a situation in which each single substitution may suffice to obtain the desired compression rate, or might be part of a more complex decision mechanism of the complete subtitling system. We thus measure the resulting accuracy of subtitles created by applying the best scoring substitution example for every original sentence. This provides a macro evaluation style since we obtain a single judgment for each group of substitution examples that correspond to one original sentence.

In our dataset 25.5% of the original sentences have no correct substitution examples and for 15.5% of the sentences all substitution examples were annotated as correct. Accordingly, the (macro averaged) accuracy has a lower bound of 0.155 and upper bound of 0.745.

4.1.2 Average Precision

As a second evaluation measure we compare the *average precision* of each method over all the examples from all original sentences pooled together (a micro averaging approach). This measures the potential of a scoring method to ensure high precision for the high scoring examples and to filter out low-scoring incorrect substitutions.

Average precision is a single figure measure commonly used to evaluate a system's ranking ability (Voorhees and Harman, 1999). It is equivalent to the area under the uninterpolated recall-precision curve, defined as follows:

$$\text{average precision} = \frac{\sum_{i=1}^{N} \mathrm{P}(i)T(i)}{\sum_{i=1}^{N} T(i)} \qquad (2)$$
$$\mathrm{P}(i) = \frac{\sum_{k=1}^{i} T(k)}{i}$$

where N is the number of examples in the test set (797 in our case), $T(i)$ is the gold annotation (true=1, false=0) and i ranges over the examples ranked by decreasing score. An average precision of 1.0 means that the system assigned a higher score to all true examples than to any false one (perfect ranking). A lower bound of 0.26 on our test set corresponds to a system that ranks all false examples above the true ones.

4.2 Results

Figure 1 shows the accuracy and average precision results of the various models on our test set. The random baseline and corresponding significance levels were achieved by averaging multiple runs of a system that assigned random scores. As can be seen in the figures, the models' behavior seems to be consistent in both evaluation measures.

Overall, the distributional similarity based method (sim) performs much better than the other methods. In particular, Lin's similarity also performs better than semcor, the other context-independent model. Generally, the context independent models perform better than the contextual ones. Between the two contextual models, nntr is superior to Bayes. In fact the Bayes model is not significantly better than random scoring.

4.3 Analysis and Discussion

When analyzing the data we identified several reasons why some of the WordNet substitutions were judged as false. In some cases the source word as appearing in the original sentence is not in a sense for which it is a synonym of the target word. For example, in many situations the word *answer* is in the sense of a statement that is made in reply to a question or request. In such cases, such as in example 2 from Table 1, *answer* can be successfully replaced with *reply* yielding a substitution which conveys the original meaning. However, in situations such as in example 1 the word *answer* is in the sense of a general solution and cannot be replaced with *reply*. This is also the case in examples 4 and 5 in which *subject* does not appear in the sense of topic or theme.

Having an inappropriate sense, however, is not the only reason for incorrect substitutions. In example 8 *approach* appears in a sense which is synonymous with attack and in example 9 *problem* appears in a sense which is synonymous with a quite uncommon use of the word job. Nevertheless, these substitutions were judged as unacceptable since the desired sense of the target word after the substitution is not very clear from the context. In many other cases, such as in example 7, though semantically correct, the substitution was judged as incorrect due to stylistic considerations.

Finally, there are cases, such as in example 6 in which the source word is part of a collocation and cannot be replaced with semantically equivalent words.

When analyzing the mistakes of the distributional similarity method it seems as if many were not necessarily due to the method itself but rather to implementation issues. The online source we used contains only the top most similar words for any word. In many cases substitutions were assigned a score of zero since they were not listed among the top scoring similar words in the database. Furthermore, the corpus that was used for training the similarity scores was news articles in American English spelling and does not always supply good scores to words of British spelling in our BBC dataset (e.g. analyse, behavioural, etc.).

The similarity based method seems to perform better than the SemCor based method since, as noted above, even when the source word is in the appropriate sense it not necessarily substitutable with the target. For this reason we hypothesize that applying Word Sense Disambiguation (WSD) methods to

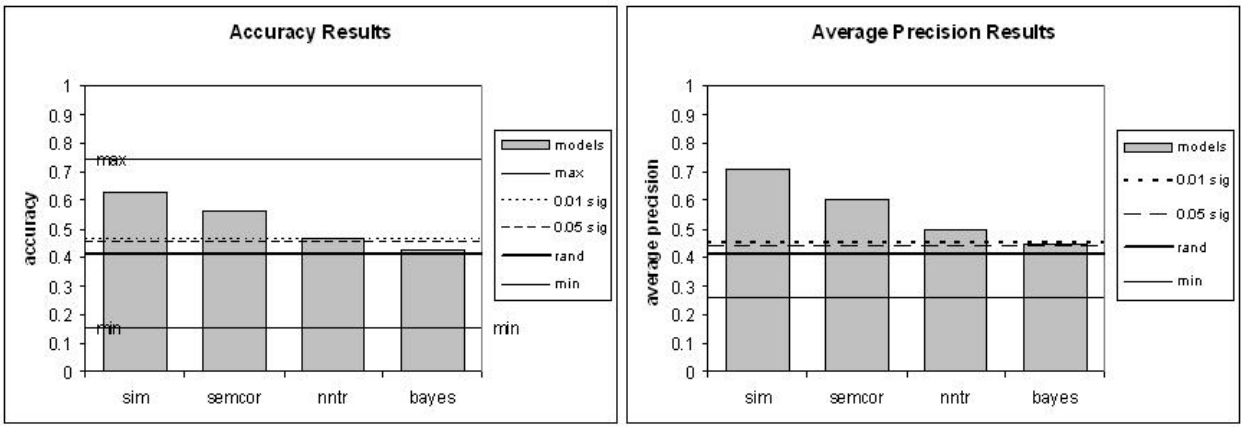


Figure 1: Accuracy and Average Precision Results

classify the specific WordNet sense of the source and target words may have only a limited impact on performance.

Overall, context independent models seem to perform relatively well since many candidate synonyms are a priori not substitutable. This demonstrates that such models are able to filter out many quirky WordNet synonyms, such as problem and job.

Fitness to the sentence context seems to be a less frequent factor and not that trivial to model. Local context (adjacent words) seems to play more of a role than the broader sentence context. However, these two types of contexts were not distinguished in the bag-of-words representations of the two contextual methods that we examined. It will be interesting to investigate in future research using different feature types for local and global context, as commonly done for Word Sense Disambiguation (WSD). Yet, it would still remain a challenging task to correctly distinguish, for example, the contexts for which *answer* is substitutable by *reply* (as in example 2) from contexts in which it is not (as in example 1).

So far we have investigated separately the performance of context independent and contextual models. In fact, the accuracy performance of the (context independent) sim method is not that far from the upper bound, and the analysis above indicated a rather small potential for improvement by incorporating information from a contextual method. Yet, there is still a substantial room for improvement in the ranking quality of this model, as measured by average precision, and it is possible that a smart combination with a high-quality contextual model would yield better performance. In particular, we would expect that a good contextual model will identify the cases in which for potentially good synonyms pair, the source word appears in a sense that is not substitutable with the target, such as in examples 1, 4 and 5 in Table 1. Investigating better contextual models and their optimal combination with context independent models remains a topic for future research.

5 Conclusion

This paper investigated an isolated setting of the lexical substitution task, which has typically been embedded in larger systems and not evaluated directly. The setting allowed us to analyze different types of state of the art models and their behavior with respect to characteristic sub-cases of the problem.

The major conclusion that seems to arise from our experiments is the effectiveness of combining a knowledge based thesaurus such as WordNet with distributional statistical information such as (Lin, 1998), overcoming the known deficiencies of each method alone. Furthermore, modeling the a priori substitution likelihood captures the majority of cases in the evaluated setting, mostly because WordNet provides a rather noisy set of substitution candidates. On the other hand, successfully incorporating local and global contextual information, as similar to WSD methods, remains a challenging task for future research. Overall, scoring lexical substitutions

is an important component in many applications and we expect that our findings are likely to be broadly applicable.

References

[Budanitsky and Hirst2001] Alexander Budanitsky and Graeme Hirst. 2001. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources: Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 29–34.

[Burnard1995] Lou Burnard. 1995. *Users Reference Guide for the British National Corpus*. Oxford University Computing Services, Oxford.

[Daelemans et al.2004] Walter Daelemans, Anja Höthker, and Erik Tjong Kim Sang. 2004. Automatic sentence simplification for subtitling in dutch and english. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1045–1048.

[Dagan et al.2005] Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment.

[Geffet and Dagan2005] Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 107–114, Ann Arbor, Michigan, June. Association for Computational Linguistics.

[Glickman et al.2005] Oren Glickman, Ido Dagan, and Moshe Koppel. 2005. A probabilistic classification approach for lexical textual entailment. In *AAAI*, pages 1050–1055.

[Grefenstette1998] Gregory Grefenstette. 1998. Producing Intelligent Telegraphic Text Reduction to Provide an Audio Scanning Service for the Blind. pages 111–117, Stanford, CA, March.

[Harris1968] Zelig Harris. 1968. *Mathematical Structures of Language*. New York: Wiley.

[Hori et al.2002] Chiori Hori, Sadaoki Furui, Rob Malkin, Hua Yu, and Alex Waibel. 2002. Automatic speech summarization applied to english broadcast news speech. volume 1, pages 9–12.

[Jing and McKeown1999] Hongyan Jing and Kathleen R. McKeown. 1999. The decomposition of human-written summary sentences. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 129–136, New York, NY, USA. ACM Press.

[Keller and Bengio2005] Mikaela Keller and Samy Bengio. 2005. A neural network for text representation. In Wodzisaw Duch, Janusz Kacprzyk, and Erkki Oja, editors, *Artificial Neural Networks: Biological Inspirations ICANN 2005: 15th International Conference, Warsaw, Poland, September 11-15, 2005. Proceedings, Part II*, volume 3697 / 2005 of *Lecture Notes in Computer Science*, page p. 667. Springer-Verlag GmbH.

[Knight and Marcu2002] Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artif. Intell.*, 139(1):91–107.

[Landis and Koch1997] J. R. Landis and G. G. Koch. 1997. The measurements of observer agreement for categorical data. *Biometrics*, 33:159–174.

[Lin1998] Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774, Morristown, NJ, USA. Association for Computational Linguistics.

[McCarthy et al.2004] Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant senses in untagged text. In *ACL*, pages 280–288, Morristown, NJ, USA. Association for Computational Linguistics.

[Miller et al.1993] George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *HLT '93: Proceedings of the workshop on Human Language Technology*, pages 303–308, Morristown, NJ, USA. Association for Computational Linguistics.

[Piperidis et al.2004] Stelios Piperidis, Iason Demiros, Prokopis Prokopidis, Peter Vanroose, Anja Höthker, Walter Daelemans, Elsa Sklavounou, Manos Konstantinou, and Yannis Karavidas. 2004. Multimodal multilingual resources in the subtitling process. In *Proceedings of the 4th International Language Resources and Evaluation Conference (LREC 2004)*, Lisbon.

[Voorhees and Harman1999] Ellen M. Voorhees and Donna Harman. 1999. Overview of the seventh text retrieval conference. In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication.

[Voorhees1994] Ellen M. Voorhees. 1994. Query expansion using lexical-semantic relations. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61–69, New York, NY, USA. Springer-Verlag New York, Inc.

Semantic Role Recognition using Kernels on Weighted Marked Ordered Labeled Trees

Jun'ichi Kazama and **Kentaro Torisawa**
Japan Advanced Institute of Science and Technology (JAIST)
Asahidai 1-1, Nomi, Ishikawa, 923-1292 Japan
{kazama, torisawa}@jaist.ac.jp

Abstract

We present a method for recognizing semantic role arguments using a kernel on weighted marked ordered labeled trees (the WMOLT kernel). We extend the kernels on marked ordered labeled trees (Kazama and Torisawa, 2005) so that the mark can be weighted according to its importance. We improve the accuracy by giving more weights on subtrees that contain the predicate and the argument nodes with this ability. Although Kazama and Torisawa (2005) presented fast training with tree kernels, the slow classification during runtime remained to be solved. In this paper, we give a solution that uses an efficient DP updating procedure applicable in argument recognition. We demonstrate that the WMOLT kernel improves the accuracy, and our speed-up method makes the recognition more than 40 times faster than the naive classification.

1 Introduction

Semantic role labeling (SRL) is a task that recognizes the arguments of a predicate (verb) in a sentence and assigns the correct role to each argument. As this task is recognized as an important step after (or the last step of) syntactic analysis, many studies have been conducted to achieve accurate semantic role labeling (Gildea and Jurafsky, 2002; Moschitti, 2004; Hacioglu et al., 2004; Punyakanok et al., 2004; Pradhan et al., 2005a; Pradhan et al., 2005b; Toutanova et al., 2005).

Most of the studies have focused on machine learning because of the availability of standard datasets, such as PropBank (Kingsbury and Palmer, 2002). Naturally, the usefulness of parse trees in this task can be anticipated. For example, the recent CoNLL 2005 shared task (Carreras and Màrquez, 2005) provided parse trees for use and their usefulness was ensured. Most of the methods heuristically extract features from parse trees, and from other sources, and use them in machine learning methods based on feature vector representation. As a result, these methods depend on feature engineering, which is time-consuming.

Tree kernels (Collins and Duffy, 2001; Kashima and Koyanagi, 2002) have been proposed to directly handle trees in kernel-based methods, such as SVMs (Vapnik, 1995). Tree kernels calculate the similarity between trees, taking into consideration all of the subtrees, and, therefore there is no need for such feature engineering.

Moschitti and Bejan (2004) extensively studied tree kernels for semantic role labeling. However, they reported that they could not successfully build an accurate argument recognizer, although the role assignment was improved. Although Moschitti et al. (2005) reported on argument recognition using tree kernels, it was a preliminary evaluation because they used oracle parse trees.

Kazama and Torisawa (2005) proposed a new tree kernel for node relation labeling, as which SRL can be cast. This kernel is defined on marked ordered labeled trees, where a node can have a mark to indicate the existence of a relation. We refer to this kernel as the MOLT kernel. Compared to (Moschitti and Bejan, 2004) where tree fragments are heuristically extracted before applying tree kernels, the MOLT kernel is general and desirable since it does not require such fragment extraction. However, the evaluation conducted by Kazama and Torisawa (2005) was limited to preliminary experiments for role assignment. In this study, we first evaluated the performance of the MOLT kernel for argument recognition, and found that the MOLT kernel cannot achieve a high accuracy if used in its original form.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 53–60, New York City, June 2006.

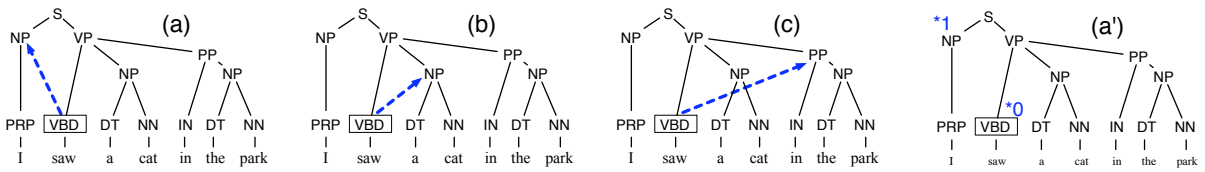


Figure 1: (a)-(c): Argument recognition as node relation recognition. (a'): relation (a) represented as marked ordered tree.

Therefore, in this paper we propose a modification of the MOLT kernel, which greatly improves the accuracy. The problem with the original MOLT kernel is that it treats subtrees with one mark, i.e., those including only the argument or the predicate node, and subtrees with two marks, i.e., those including both the argument and the predicate nodes equally, although the latter is likely to be more important for distinguishing difficult arguments. Thus, we modified the MOLT kernel so that the marks can be weighted in order to give large weights to the subtrees with many marks. We call the modified kernel the WMOLT kernel (the kernel on *weighted marked ordered labeled trees*). We show that this modification greatly improves the accuracy when the weights for marks are properly tuned.

One of the issues that arises when using tree kernels is time complexity. In general, tree kernels can be calculated in $O(|T_1||T_2|)$ time, where $|T_i|$ is the number of nodes in tree T_i, using dynamic programming (DP) procedures (Collins and Duffy, 2001; Kashima and Koyanagi, 2002). However, this cost is not negligible in practice. Kazama and Torisawa (2005) proposed a method that drastically speeds up the calculation during training by converting trees into efficient vectors using a tree mining algorithm. However, the slow classification during runtime remained an open problem.

We propose a method for speeding up the runtime classification for argument recognition. In argument recognition, we determine whether a node is an argument or not for all the nodes in a tree . This requires a series of calculations between a support vector tree and a tree with slightly different marking. By exploiting this property, we can efficiently update DP cells to obtain the kernel value with less computational cost.

In the experiments, we demonstrated that the WMOLT kernel drastically improved the accuracy and that our speed-up method enabled more than 40 times faster argument recognition. Despite these successes, the performance of our current system is $F_1 = 78.22$ on the CoNLL 2005 evaluation set when using the Charniak parse trees, which is far worse than the state-of-the-art system. We will present possible reasons and future directions.

2 Semantic Role Labeling

Semantic role labeling (SRL) recognizes the arguments of a given predicate and assigns the correct role to each argument. For example, the sentence "I saw a cat in the park" will be labeled as follows with respect to the predicate "see".

[A0 I] [V saw] [A1 a cat] [AM-LOC in the park]

In the example, A0, A1, and AM-LOC are the roles assigned to the arguments. In the CoNLL 2005 dataset, there are the numbered arguments (AX) whose semantics are predicate dependent, the adjuncts (AM-X), and the references (R-X) for relative clauses.

Many previous studies employed two-step SRL methods, where (1) we first recognize the arguments, and then (2) classify the argument to the correct role. We also assume this two-step processing and focus on the argument recognition.

Given a parse tree, argument recognition can be cast as the classification of tree nodes into two classes, "ARG" and "NO-ARG". Then, we consider the words (a phrase) that are the descendants of an "ARG" node to be an argument. Since arguments are defined for a given predicate, this classification is the recognition of a relation between the predicate and tree nodes. Thus, we want to build a binary classifier that returns a +1 for correct relations and a -1 for incorrect relations. For the above example, the classifier will output a +1 for the relations indicated by (a), (b), and (c) in Figure 1 and a -1 for the relations between the predicate node and other nodes.

Since the task is the classification of trees with node relations, tree kernels for usual ordered labeled trees, such as those proposed by Collins and Duffy (2001) and Kashima and Koyanagi (2002), are not useful. Kazama and Torisawa (2005) proposed to represent a node relation in a tree as a marked ordered labeled tree and presented a kernel for it (MOLT kernel). We adopted the MOLT kernel and extend it for accurate argument recognition.

3 Kernels for Argument Recognition

3.1 Kernel-based classification

Kernel-based methods, such as support vector machines (SVMs) (Vapnik, 1995), consider a mapping $\Phi(x)$ that maps the object x into a, (usually high-dimensional), feature space and learn a classifier in this space. A kernel function $K(x_i, x_j)$ is a function that calculates the inner product $\langle \Phi(x_i), \Phi(x_j) \rangle$ in the feature space without explicitly computing $\Phi(x)$, which is sometimes intractable. Then, any classifier that is represented by using only the inner products between the vectors in a feature space can be rewritten using the kernel function. For example, an SVM classifier has the form:

$$f(x) = \sum_i \alpha_i K(x_i, x) + b,$$

where α_i and b are the parameters learned in the training. With kernel-based methods, we can construct a powerful classifier in a high-dimensional feature space. In addition, objects x do not need to be vectors as long as a kernel function is defined (e.g., x can be strings, trees, or graphs).

3.2 MOLT kernel

A marked ordered labeled tree (Kazama and Torisawa, 2005) is an ordered labeled tree in which each node can have a mark in addition to a label. We can encode a k-node relation by using k distinct marks. In this study, we determine an argument node without considering other arguments of the same predicate, i.e., we represent an argument relation as a two-node relation using two marks. For example, the relation (a) in Figure 1 can be represented as the marked ordered labeled tree (a').[1]

[1]Note that we use mark *0 for the predicate node and mark *1 for the argument node.

Table 1: Notations for MOLT kernel.

- n_i denotes a node of a tree. In this paper, n_i is an ID assigned in the post-order traversal.
- $|T_i|$ denotes the number of nodes in tree T_i.
- $l(n_i)$ returns the label of node n_i.
- $m(n_i)$ returns the mark of node n_i. If n_i has no mark, $m(n_i)$ returns the special mark no-mark.
- $marked(n_i)$ returns $true$ iff $m(n_i)$ is not no-mark.
- $nc(n_i)$ is the number of children of node n_i.
- $ch_k(n_i)$ is the k-th child of node n_i.
- $pa(n_i)$ is the parent of node n_i.
- $root(T_i)$ is the root node of T_i
- $n_i \succeq n_j$ means that n_i is an elder sister of n_j.

Kazama and Torisawa (2005) presented a kernel on marked ordered trees (the MOLT kernel), which is defined as:[2]

$$K(T_1, T_2) = \sum_{i=1}^{E} W(S_i) \cdot \#_{S_i}(T_1) \cdot \#_{S_i}(T_2),$$

where S_i is a possible subtree and $\#_{S_i}(T_j)$ is the number of times S_i is *included* in T_j. The mapping corresponding to this kernel is $\Phi(T) = (\sqrt{W(S_1)}\#_{S_1}(T), \cdots, \sqrt{W(S_E)}\#_{S_E}(T))$, which maps the tree into the feature space of all the possible subtrees.

The tree inclusion is defined in many ways. For example, Kashima and Koyanagi (2002) presented the following type of inclusion.

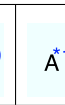
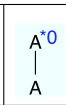
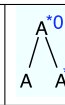
1 DEFINITION *S is included in T iff there exists a one-to-one function ψ from a node of S to a node of T, such that (i) $pa(\psi(n_i)) = \psi(pa(n_i))$, (ii) $\psi(n_i) \succeq \psi(n_j)$ iff $n_i \succeq n_j$, , and (iii) $l(\psi(n_i)) = l(n_i)$ (and $m(\psi(n_i)) = m(n_i)$ in the MOLT kernel).*

See Table 1 for the meaning of each function. This definition means that any subtrees preserving the parent-child relation, the sibling relation, and label-marks, are allowed. In this paper, we employ this definition, since Kazama and Torisawa (2005) reported that the MOLT kernel with this definition has a higher accuracy than one with the definition presented by Collins and Duffy (2001).

$W(S_i)$ is the weight of subtree S_i. The weighting in Kazama and Torisawa (2005) is written as fol-

[2]This notation is slightly different from (Kazama and Torisawa, 2005).

Table 2: Example of subtree inclusion and subtree weights. The last row shows the weights for WMOLT kernel.

T	included subtrees					
						
$W(S_i)$	0	λ	λ	λ^2	λ^2	λ^3
$W(S_i)$	0	$\lambda\gamma$	$\lambda\gamma$	$\lambda^2\gamma$	$\lambda^2\gamma^2$	$\lambda^3\gamma^2$

lows.

$$W(S_i) = \begin{cases} \lambda^{|S_i|} & \text{if } marked(S_i), \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where $marked(S_i)$ returns $true$ iff $marked(n_i) = true$ for at least one node in tree S_i. By this weighting, only the subtrees with at least one mark are considered. The idea behind this is that subtrees having no marks are not useful for relation recognition or labeling. λ ($0 \leq \lambda \leq 1$) is a factor to prevent the kernel values from becoming too large, which has been used in previous studies (Collins and Duffy, 2001; Kashima and Koyanagi, 2002).

Table 2 shows an example of subtree inclusion and the weights given to each included subtree. Note that the subtrees are treated differently when the markings are different, even if the labels are the same.

Although the dimension of the feature space is exponential, tree kernels can be calculated in $O(|T_1||T_2|)$ time using dynamic programming (DP) procedures (Collins and Duffy, 2001; Kashima and Koyanagi, 2002). The MOLT kernel also has an $O(|T_1||T_2|)$ DP procedure (Kazama and Torisawa, 2005).

3.3 WMOLT kernel

Although Kazama and Torisawa (2005) evaluated the MOLT kernel for SRL, the evaluation was only on the role assignment task and was preliminary. We evaluated the MOLT kernel for argument recognition, and found that the MOLT kernel cannot achieve a high accuracy for argument recognition.

The problem is that the MOLT kernel treats subtrees with one mark and subtrees with two marks equally, although the latter seems to be more important in distinguishing difficult arguments.

Consider the sentence, "He said industry should build plants". For "say", we have the following labeling.

[A0 He] [V said] [A1 industry should build plants]

On the other hand, for "build", we have

He said [A0 industry] [AM-MOD should] [V build] [A1 plants].

As can be seen, "he" is the A0 argument of "say", but not an argument of "build". Thus, our classifier should return a +1 for the tree where "he" is marked when the predicate is "say", and a -1 when the predicate is "build". Although the subtrees around the node for "say" and "build" are different, the subtrees around the node for "he" are identical for both cases. If "he" is often the A0 argument in the corpus, it is likely that the classifier returns a +1 even for "build". Although the subtrees containing both the predicate and the argument nodes are considered in the MOLT kernel, they are given relatively small weights by Eq. (1), since such subtrees are large.

Thus, we modify the MOLT kernel so that the mark can be weighted according to its importance and the more marks the subtrees contain, the more weights they get. The modification is simple. We change the definition of $W(S_i)$ as follows.

$$W(S_i) = \begin{cases} \lambda^{|S_i|} \prod_{n_i \in S_i} \gamma(m(n_i)) & \text{if } marked(S_i), \\ 0 & \text{otherwise,} \end{cases}$$

where $\gamma(m)$ (≥ 1) is the weight of mark m. We call a kernel with this weight the WMOLT kernel. In this study, we assume $\gamma(\text{no-mark}) = 1$ and $\gamma(*0) = \gamma(*1) = \gamma$. Then, the weight is simplified as follows.

$$W(S_i) = \begin{cases} \lambda^{|S_i|}\gamma^{\#_m(S_i)} & \text{if } marked(S_i), \\ 0 & \text{otherwise,} \end{cases}$$

where $\#_m(S_i)$ is the number of marked nodes in S_i. The last row in Table 2 shows how the subtree weights change by introducing this mark weighting.

For the WMOLT kernel, we can derive $O(|T_1||T_2|)$ DP procedure by slightly modifying the procedure presented by Kazama and Torisawa (2005). The method for speeding up training described in Kazama and Torisawa (2005) can also be applied with a slight modification.

Algorithm 3.1: WMOLT-KERNEL(T_1, T_2)

for $n_1 \leftarrow 1$ **to** $|T_1|$ **do** // nodes are ordered by the post-order traversal
  $m \leftarrow marked(n_1)$
  for $n_2 \leftarrow 1$ **to** $|T_2|$ **do** // actually iterate only on n_2 with $l(n_1) = l(n_2)$
(A)
  - **if** $l(n_1) \neq l(n_2)$ **or** $m(n_1) \neq m(n_2)$ **then**
    $C(n_1, n_2) \leftarrow 0$ $C^r(n_1, n_2) \leftarrow 0$
  - **else if** n_1 and n_2 are leaf nodes **then**
    if m **then** $C(n_1, n_2) \leftarrow \lambda \cdot \gamma$; $C^r(n_1, n_2) \leftarrow \lambda \cdot \gamma$ **else** $C(n_1, n_2) \leftarrow \lambda$; $C^r(n_1, n_2) \leftarrow 0$
  - **else**
    $S(0, j) \leftarrow 1$, $S(i, 0) \leftarrow 1$ $(i \in [0, nc(n_1)], j \in [0, nc(n_2)])$
    if m **then** $S^r(0, j) \leftarrow 1$, $S^r(i, 0) \leftarrow 1$ **else** $S^r(0, j) \leftarrow 0$, $S^r(i, 0) \leftarrow 0$
    for $i \leftarrow 1$ **to** $nc(n_1)$ **do**
      for $j \leftarrow 1$ **to** $nc(n_2)$ **do**
        $S(i, j) \leftarrow S(i-1, j) + S(i, j-1) - S(i-1, j-1) + S(i-1, j-1) \cdot C(ch_i(n_1), ch_j(n_2))$
        $S^r(i, j) \leftarrow S^r(i-1, j) + S^r(i, j-1) - S^r(i-1, j-1) + S^r(i-1, j-1) \cdot C(ch_i(n_1), ch_j(n_2))$
          $+S(i-1, j-1) \cdot C^r(ch_i(n_1), ch_j(n_2)) - S^r(i-1, j-1) \cdot C^r(ch_i(n_1), ch_j(n_2))$
    if m **then** $C(n_1, n_2) \leftarrow \lambda \cdot \gamma \cdot S(nc(n_1), nc(n_2))$ **else** $C(n_1, n_2) \leftarrow \lambda \cdot S(nc(n_1), nc(n_2))$
    if m **then** $C^r(n_1, n_2) \leftarrow \lambda \cdot \gamma \cdot S^r(nc(n_1), nc(n_2))$ **else** $C^r(n_1, n_2) \leftarrow \lambda \cdot S^r(nc(n_1), nc(n_2))$

return $(\sum_{n_1=1}^{|T_1|} \sum_{n_2=1}^{|T_2|} C^r(n_1, n_2))$

We describe this DP procedure in some detail. The key is the use of two DP matrices of size $|T_1| \times |T_2|$. The first is $C(n_1, n_2)$ defined as:

$$C(n_1, n_2) \equiv \sum_{S_i} W'(S_i) \cdot \#_{S_i}(T_1 \triangle n_1) \cdot \#_{S_i}(T_2 \triangle n_2),$$

where $\#_{S_i}(T_j \triangle n_k)$ represents the number of times subtree S_i is included in tree T_j with $\psi(root(S_i)) = n_k$. $W'(S_i)$ is defined as $W'(S_i) = \lambda^{|S_i|} \gamma^{\#_m(S_i)}$. This means that this matrix records the values that ignore whether $marked(S_i) = true$ or not. The second is $C^r(n_1, n_2)$ defined as:

$$C^r(n_1, n_2) \equiv \sum_{S_i} W(S_i) \cdot \#_{S_i}(T_1 \triangle n_1) \cdot \#_{S_i}(T_2 \triangle n_2).$$

With these matrices, the kernel is calculated as:

$$K(T_1, T_2) = \sum_{n_1 \in T_1} \sum_{n_2 \in T_2} C^r(n_1, n_2).$$

$C(n_1, n_2)$ and $C^r(n_1, n_2)$ are calculated recursively, starting from the leaves of the trees. The recursive procedure is shown in Algorithm 3.1. See also Table 1 for the meaning of the functions used.

4 Fast Argument Recognition

We use the SVMs for the classifiers in argument recognition in this study and describe the fast classification method based on SVMs.[3] We denote a marked ordered labeled tree where node n_k of an ordered labeled tree U is marked by mark X, n_l by Y, and so on, by $U@\{n_k = X, n_l = Y, \dots\}$.

[3]The method can be applied to a wide range of kernel-based methods that have the same structure as SVMs.

Algorithm 4.1: CALCULATE-T(U, T_j)

procedure FAST-UPDATE(n_k)
  $diff \leftarrow 0$, $m(n_k) \leftarrow$ *1, $\mathcal{U} \leftarrow \phi$
  for $n_2 \leftarrow 1$ **to** $|T_j|$ **do** $change(n_2) \leftarrow$ **true**
  $n_1 \leftarrow n_k$
  while $n_1 \neq nil$ **do**
    for $n_2 \leftarrow 1$ **to** $|T_j|$ **do**
      // actually iterate only on n_2 with $l(pa(n_1)) = l(n_2)$
      $nchange(n_2) \leftarrow$ **false**
    for $n_2 \leftarrow 1$ **to** $|T_j|$ **do**
      // actually iterate only on n_2 with $l(n_1) = l(n_2)$
      if $change(n_2)$ **then**
        $pre \leftarrow C^r(n_1, n_2)$, $\mathcal{U} \leftarrow \mathcal{U} \cup (n_1, n_2)$
        update $C(n_1, n_2)$ and $C^r(n_1, n_2)$
          using (A) of Algorithm 3.1
        $diff$ += $(C^r(n_1, n_2) - pre)$
        if $pa(n_2) \neq nil$ **then** $nchange(pa(n_2)) \leftarrow$ **true**
    $n_1 \leftarrow pa(n_1)$, $change \leftarrow nchange$
  for $(n_1, n_2) \in \mathcal{U}$ **do** //restore DP cells
    $C(n_1, n_2) \leftarrow C'(n_1, n_2)$, $C^r(n_1, n_2) \leftarrow C^{r\prime}(n_1, n_2)$
  $m(n_k) \leftarrow$ no-mark
  return ($diff$)

main
  $m(n_v) \leftarrow *0$, $k \leftarrow$ WMOLT-KERNEL(U, T_j)
  $C'(n_1, n_2) \leftarrow C(n_1, n_2)$, $C^{r\prime}(n_1, n_2) \leftarrow C^r(n_1, n_2)$
  for $n_k \leftarrow 1$ **to** $|U|$ **do** $(n_k \neq n_v)$
    $diff \leftarrow$ FAST-UPDATE(n_k), $t(n_k) \leftarrow k + diff$

Given a sentence represented by tree U and the node for the target predicate n_v, the argument recognition requires the calculation of:

$$s(n_k) = \sum_{T_j \in \mathcal{SV}} \alpha_j K(U@\{n_v = *0, n_k = *1\}, T_j) + b, \quad (2)$$

for all $n_k \in U$ $(\neq n_v)$, where $\mathcal{SV}$ represents the support vectors. Naively, this requires $O(|U| \times |\mathcal{SV}| \times |U||T_j|)$ time, which is rather costly in practice.

However, if we exploit the fact that $U@\{n_v = *0, n_k = *1\}$ is different from $U@\{n_v = *0\}$ at one node, we can greatly speed up the above calculation. At first, we calculate $K(U@\{n_v = *0\}, T_j)$ using the DP procedure presented in the previous section, and then calculate $K(U@\{n_v = *0, n_k = *1\}, T_j)$ using a more efficient DP that updates only the values of the necessary DP cells of the first DP. More specifically, we only need to update the DP cells involving the ancestor nodes of n_k.

Here we show the procedure for calculating $t(n_k) = K(U@\{n_v = *0, n_k = *1\}, T_j)$ for all n_k for a given support vector T_j, which will suffice for calculating $s(n_k)$. Algorithm 4.1 shows the procedure. For each n_k, this procedure updates at most (n_k's depth) $\times$ $|T_j|$ cells, which is much less than $|U| \times |T_j|$ cells. In addition, when updating the cells for (n_1, n_2), we only need to update them when the cells for any child of n_2 have been updated in the calculation of the cells for the children of n_1. To achieve this, $change(n_2)$ in the algorithm stores whether the cells of any child of n_2 have been updated. This technique will also reduce the number of updated cells.

5 Non-overlapping Constraint

Finally, in argument recognition, there is a strong constraint that the arguments for a given predicate do not overlap each other. To enforce this constraint, we employ the approach presented by Toutanova et al. (2005). Given the local classification probability $p(n_k = X_k) \quad (X_k \in \{\text{ARG}, \text{NO-ARG}\})$, this method finds the assignment that maximizes $\prod_k p(n_k = X_k)$ while satisfying the above non-overlapping constraint, by using a dynamic programming procedure. Since the output of SVMs is not a probability value, in this study we obtain the probability value by converting the output from the SVM, $s(n_k)$, using the sigmoid function:[4]

$$p(n_k = \text{ARG}) = 1/(1 + \exp(-s(n_k))).$$

6 Evaluation

6.1 Setting

For our evaluation we used the dataset provided for the CoNLL 2005 SRL shared task (www.lsi.upc.edu/~srlconll). We used only the training part and divided it into our training, development, and test sets (23,899, 7,966, and 7,967 sentences, respectively). We used the outputs of the Charniak parser provided with the dataset. We also used POS tags, which were also provided, by inserting the nodes labeled by POS tags above the word nodes. The words were downcased.

We used TinySVM[5] as the implementation of the SVMs, adding the WMOLT kernel. We normalized the kernel as: $K(T_i, T_j)/\sqrt{K(T_i, T_i) \times K(T_j, T_j)}$.

To train the classifiers, for a positive example we used the marked ordered labeled tree that encodes an argument in the training set. Although nodes other than the argument nodes were potentially negative examples, we used 1/5 of these nodes that were randomly-sampled, since the number of such nodes is so large that the training cannot be performed in practice. Note that we ignored the arguments that do not match any node in the tree (the rate of such arguments was about 3.5% in the training set).

6.2 Effect of mark weighting

We first evaluated the effect of the mark weighting of the WMOLT kernel. For several fixed γ, we tuned λ and the soft-margin constant of the SVM, C, and evaluated the recognition accuracy. We tested 30 different values of $C \in [0.1 \ldots 500]$ for each $\lambda \in [0.05, 0.1, 0.15, 0.2, 0.25, 0.3]$. The tuning was performed using the method for speeding up the training with tree kernels described by Kazama and Torisawa (2005). We conducted the above experiment for several training sizes.

Table 3 shows the results. This table shows the best setting of λ and C, the performance on the development set with the best setting, and the performance on the test set. The performance is shown in the F_1 measure. Note that we treated the region labeled C-k in the CoNLL 2005 dataset as an independent argument.

We can see that the mark weighting greatly improves the accuracy over the original MOLT kernel (i.e., $\gamma = 1$). In addition, we can see that the best setting for γ is somewhere around $\gamma = 4,000$. In this experiment, we could only test up to 1,000 sentences due to the cost of SVM training, which were

[4] Parameter fitting (Platt, 1999) is not performed.

[5] chasen.org/~taku/software/TinySVM

Table 3: Effect of γ in mark weighting of WMOLT kernel.

	training size (No. of sentences)											
	250			500			700			1,000		
γ	setting (λ,C)	dev (F_1)	test (F_1)	setting (λ,C)	dev (F_1)	test (F_1)	setting (λ,C)	dev (F_1)	test (F_1)	setting (λ,C)	dev (F_1)	test (F_1)
1	0.15, 20.50	63.66	65.13	0.2, 20.50	69.01	70.33	0.2, 20.50	72.11	73.57	0.25, 12.04	75.38	76.25
100	0.3, 12.04	80.13	80.85	0.3,500	82.25	82.98	0.3, 34.92	83.93	84.72	0.3, 3.18	85.09	85.85
1,000	0.2, 2.438	82.65	83.36	0.2, 2.438	84.80	85.45	0.2, 3.182	85.58	86.20	0.2, 7.071	86.40	86.80
2,000	0.2, 2.438	83.43	84.12	0.2, 2.438	**85.56**	**86.24**	0.2, 2.438	**86.23**	**86.80**	0.2, 12.04	86.61	87.18
4,000	0.2, 2.438	**83.87**	**84.50**	0.15, 4.15	84.94	85.61	0.15, 7.07	85.84	86.32	0.2, 12.04	**86.82**	**87.31**
4,000 (w/o)		80.81	81.41		80.71	81.51		81.86	82.33		84.27	84.63

empirically $O(L^2)$ where L is the number of training examples, regardless of the use of the speed-up method (Kazama and Torisawa, 2005), However, we can observe that the WMOLT kernel achieves a high accuracy even when the training data is very small.

6.3 Effect of non-overlapping constraint

Additionally, we observed how the accuracy changes when we do not use the method described in Section 5 and instead consider the node to be an argument when $s(n_k) > 0$. The last row in Table 3 shows the accuracy for the model obtained with $\gamma = 4,000$. We could observe that the non-overlapping constraint also improves the accuracy.

6.4 Recognition speed-up

Next, we examined the method for fast argument recognition described in Section 4. Using the classifiers with $\gamma = 4,000$, we measured the time required for recognizing the arguments for 200 sentences with the naive classification of Eq. (2) and with the fast update procedure shown in Algorithm 4.1. The time was measured using a computer with 2.2-GHz dual-core Opterons and 8-GB of RAM.

Table 4 shows the results. We can see a constant speed-up by a factor of more than 40, although the time was increased for both methods as the size of the training data increases (due to the increase in the number of support vectors).

Table 4: Recognition time (sec.) with naive classification and proposed fast update.

	training size (No. of sentences)			
	250	500	750	1,000
naive	11,266	13,008	18,313	30,226
proposed	226	310	442	731
speed-up	49.84	41.96	41.43	41.34

6.5 Evaluation on CoNLL 2005 evaluation set

To compare the performance of our system with other systems, we conducted the evaluation on the official evaluation set of the CoNLL 2005 shared task. We used a model trained using 2,000 sentences (57,547 examples) with ($\gamma = 4,000, \lambda = 0.2, C = 12.04$), the best setting in the previous experiments. This is the largest model we have successfully trained so far, and has $F_1 = 88.00$ on the test set in the previous experiments.

The accuracy of this model on the official evaluation set was $F_1 = 79.96$ using the criterion from the previous experiments where we treated a C-k argument as an independent argument. The official evaluation script returned $F_1 = 78.22$. This difference is caused because the official script takes C-k arguments into consideration, while our system cannot output C-k labels since it is just an argument recognizer. Therefore, the performance will become slightly higher than $F_1 = 78.22$ if we perform the role assignment step. However, our current system is worse than the systems reported in the CoNLL 2005 shared task in any case, since it is reported that they had $F_1 = 79.92$ to 83.78 argument recognition accuracy (Carreras and Màrquez, 2005).

7 Discussion

Although we have improved the accuracy by introducing the WMOLT kernel, the accuracy for the official evaluation set was not satisfactory. One possible reason is the accuracy of the parser. Since the Charniak parser is trained on the same set with the training set of the CoNLL 2005 shared task, the parsing accuracy is worse for the official evaluation set than for the training set. For example, the rate of the arguments that do not match any node of the parse tree is 3.93% for the training set, but 8.16% for the

evaluation set. This, to some extent, explains why our system, which achieved $F_1 = 88.00$ for our test set, could only achieved $F_1 = 79.96$. To achieve a higher accuracy, we need to make the system more robust to parsing errors. Some of the non-matching arguments are caused by incorrect treatment of quotation marks and commas. These errors seem to be solved by using simple pre-processing. Other major non-matching arguments are caused by PP attachment errors. To solve these errors, we need to explore more, such as using n-best parses and the use of several syntactic views (Pradhan et al., 2005b).

Another reason for the low accuracy is the size of the training data. In this study, we could train the SVM with 2,000 sentences (this took more than 30 hours including the conversion of trees), but this is a very small fraction of the entire training set. We need to explore the methods for incorporating a large training set within a reasonable training time. For example, the combination of small SVMs (Shen et al., 2003) is a possible direction.

The contribution of this study is not the accuracy achieved. The first contribution is the demonstration of the drastic effect of the mark weighting. We will explore more accurate kernels based on the WMOLT kernel. For example, we are planning to use different weights depending on the marks. The second contribution is the method of speeding-up argument recognition. This is of great importance, since the proposed method can be applied to other tasks where all nodes in a tree should be classified. In addition, this method became possible because of the WMOLT kernel, and it is hard to apply to Moschitti and Bejan (2004) where the tree structure changes during recognition. Thus, the architecture that uses the WMOLT kernel is promising, if we assume further progress is possible with the kernel design.

8 Conclusion

We proposed a method for recognizing semantic role arguments using the WMOLT kernel. The mark weighting introduced in the WMOLT kernel greatly improved the accuracy. In addition, we presented a method for speeding up the recognition, which resulted in more than a 40 times faster recognition. Although the accuracy of the current system is worse than the state-of-the-art system, we expect to further improve our system.

References

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *CoNLL 2005*.

M. Collins and N. Duffy. 2001. Convolution kernels for natural language. In *NIPS 2001*.

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

K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *CoNLL 2004*.

H. Kashima and T. Koyanagi. 2002. Kernels for semi-structured data. In *ICML 2002*, pages 291–298.

J. Kazama and K. Torisawa. 2005. Speeding up training with tree kernels for node relation labeling. In *EMNLP 2005*.

P. Kingsbury and M. Palmer. 2002. From treebank to propbank. In *LREC 02*.

A. Moschitti and C. A. Bejan. 2004. A semantic kernels for predicate argument classification. In *CoNLL 2004*.

A. Moschitti, B. Coppola, D. Pighin, and B. Basili. 2005. Engineering of syntactic features for shallow semantic parsing. In *ACL 2005 Workshop on Feature Enginnering for Machine Learning in Natural Language Processing*.

A. Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *ACL 2004*.

J. C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*.

S. Pradhan, K. Hacioglu, W. Ward, D. Jurafsky, and J. H. Martin. 2005a. Support vector learning for semantic argument classification. *Machine Learning*, 60(1).

S. Pradhan, W. Ward, K. Hacioglu, J. H. Martin, and D. Jurafsky. 2005b. Semantic role labeling using different syntactic views. In *ACL 2005*.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *COLING 2004*.

L. Shen, A. Sarkar, and A. K. Joshi. 2003. Using LTAG based features in parse reranking. In *EMNLP 2003*.

K. Toutanova, A. Haghighi, and C. D. Manning. 2005. Joint learning improves semantic role labeling. In *ACL 2005*.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer Verlag.

Semantic Role Labeling via Tree Kernel Joint Inference

Alessandro Moschitti, Daniele Pighin and **Roberto Basili**
Department of Computer Science
University of Rome "Tor Vergata"
00133 Rome, Italy
{moschitti,basili}@info.uniroma2.it
daniele.pighin@gmail.com

Abstract

Recent work on Semantic Role Labeling (SRL) has shown that to achieve high accuracy a joint inference on the whole predicate argument structure should be applied. In this paper, we used syntactic subtrees that span potential argument structures of the target predicate in tree kernel functions. This allows Support Vector Machines to discern between correct and incorrect predicate structures and to re-rank them based on the joint probability of their arguments. Experiments on the PropBank data show that both classification and re-ranking based on tree kernels can improve SRL systems.

1 Introduction

Recent work on Semantic Role Labeling (SRL) (Carreras and Màrquez, 2005) has shown that to achieve high labeling accuracy a joint inference on the whole predicate argument structure should be applied. For this purpose, we need to extract features from the sentence's syntactic parse tree that encodes the target semantic structure. This task is rather complex since we do not exactly know which are the syntactic clues that capture the relation between the predicate and its arguments. For example, to detect the interesting context, the modeling of syntax/semantics-based features should take into account linguistic aspects like ancestor nodes or semantic dependencies (Toutanova et al., 2004).

A viable approach to generate a large number of features has been proposed in (Collins and Duffy, 2002), where convolution kernels were used to implicitly define a tree substructure space. The selection of the relevant structural features was left to the Voted Perceptron learning algorithm. Such successful experimentation shows that tree kernels are very promising for automatic feature engineering, especially when the available knowledge about the phenomenon is limited.

In a similar way, we can model SRL systems with tree kernels to generate large feature spaces. More in detail, most SRL systems split the labeling process into two different steps: Boundary Detection (i.e. to determine the text boundaries of predicate arguments) and Role Classification (i.e. labeling such arguments with a semantic role, e.g. Arg0 or Arg1 as defined in (Kingsbury and Palmer, 2002)). The former relates to the detection of syntactic parse tree nodes associated with constituents that correspond to arguments, whereas the latter considers the boundary nodes for the assignment of the suitable label. Both steps require the design and extraction of features from parse trees. As capturing the tightly interdependent relations among a predicate and its arguments is a complex task, we can apply tree kernels on the subtrees that *span* the whole predicate argument structure to generate the feature space of all the possible subtrees.

In this paper, we apply the traditional boundary (TBC) and role (TRC) classifiers (Pradhan et al., 2005a), which are based on binary predicate/argument relations, to label all parse tree nodes corresponding to potential arguments. Then, we ex-

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 61–68, New York City, June 2006.

tract the subtrees which span the predicate-argument dependencies of such arguments, i.e. Argument Spanning Trees (ASTs). These are used in a tree kernel function to generate all possible substructures that encode n-ary argument relations, i.e. we carry out an automatic feature engineering process.

To validate our approach, we experimented with our model and Support Vector Machines for the classification of valid and invalid ASTs. The results show that this classification problem can be learned with high accuracy. Moreover, we modeled SRL as a re-ranking task in line with (Toutanova et al., 2005). The large number of complex features provided by tree kernels for structured learning allows SVMs to reach the state-of-the-art accuracy.

The paper is organized as follows: Section 2 introduces the Semantic Role Labeling based on SVMs and the tree kernel spaces; Section 3 formally defines the ASTs and the algorithm for their classification and re-ranking; Section 4 shows the comparative results between our approach and the traditional one; Section 5 presents the related work; and finally, Section 6 summarizes the conclusions.

2 Semantic Role Labeling

In the last years, several machine learning approaches have been developed for automatic role labeling, e.g. (Gildea and Jurasfky, 2002; Pradhan et al., 2005a). Their common characteristic is the adoption of attribute-value representations for predicate-argument structures. Accordingly, our basic system is similar to the one proposed in (Pradhan et al., 2005a) and it is hereby described.

We use a boundary detection classifier (for any role type) to derive the words compounding an argument and a multiclassifier to assign the roles (e.g. `Arg0` or `ArgM`) described in PropBank (Kingsbury and Palmer, 2002)). To prepare the training data for both classifiers, we used the following algorithm:

1. Given a sentence from the *training-set*, generate a full syntactic parse tree;
2. Let $\mathcal{P}$ and $\mathcal{A}$ be respectively the set of predicates and the set of parse-tree nodes (i.e. the potential arguments);
3. For each pair $\langle p, a\rangle \in \mathcal{P} \times \mathcal{A}$:

   - extract the feature representation set, $F_{p,a}$;
   - if the subtree rooted in a covers exactly the words of one argument of p, put $F_{p,a}$ in the T^+ set (positive examples), otherwise put it in the T^- set (negative examples).

The outputs of the above algorithm are the T^+ and T^- sets. These sets can be directly used to train a boundary classifier (e.g. an SVM). Regarding the argument type classifier, a binary labeler for a role r (e.g. an SVM) can be trained on the T^+_r, i.e. its positive examples and T^-_r, i.e. its negative examples, where $T^+ = T^+_r \cup T^-_r$, according to the ONE-vs-ALL scheme. The binary classifiers are then used to build a general role multiclassifier by simply selecting the argument associated with the maximum among the SVM scores.

Regarding the design of features for predicate-argument pairs, we can use the attribute-values defined in (Gildea and Jurasfky, 2002) or tree structures (Moschitti, 2004). Although we focus on the latter approach, a short description of the former is still relevant as they are used by TBC and TRC. They include the *Phrase Type*, *Predicate Word*, *Head Word*, *Governing Category*, *Position* and *Voice* features. For example, the *Phrase Type* indicates the syntactic type of the phrase labeled as a predicate argument and the *Parse Tree Path* contains the path in the parse tree between the predicate and the argument phrase, expressed as a sequence of nonterminal labels linked by direction (up or down) symbols, e.g. `V` ↑ `VP` ↓ `NP`.

A viable alternative to manual design of syntactic features is the use of tree-kernel functions. These implicitly define a feature space based on all possible tree substructures. Given two trees T_1 and T_2, instead of representing them with the whole fragment space, we can apply the kernel function to evaluate the number of common fragments.

Formally, given a tree fragment space $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$, the indicator function $I_i(n)$ is equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. A tree-kernel function over t_1 and t_2 is $K_t(t_1, t_2) = \sum_{n_1 \in N_{t_1}} \sum_{n_2 \in N_{t_2}} \Delta(n_1, n_2)$, where N_{t_1} and N_{t_2} are the sets of the t_1's and t_2's nodes, respectively. In turn $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \lambda^{l(f_i)} I_i(n_1) I_i(n_2)$, where $0 \leq \lambda \leq 1$ and $l(f_i)$ is the height of the subtree f_i. Thus $\lambda^{l(f_i)}$ assigns a lower weight to larger frag-

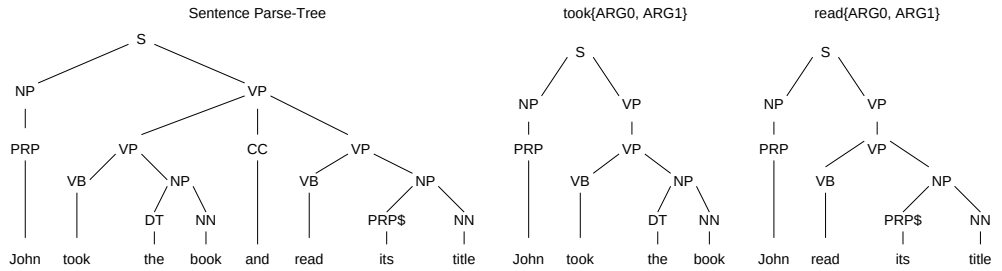


Figure 1: A sentence parse tree with two argument spanning trees (ASTs)

ments. When $\lambda = 1$, Δ is equal to the number of common fragments rooted at nodes n_1 and n_2. As described in (Collins and Duffy, 2002), Δ can be computed in $O(|N_{t_1}| \times |N_{t_2}|)$.

3 Tree kernel-based classification of Predicate Argument Structures

Traditional semantic role labeling systems extract features from pairs of nodes corresponding to a predicate and one of its argument, respectively. Thus, they focus on only binary relations to make classification decisions. This information is poorer than the one expressed by the whole predicate argument structure. As an alternative we can select the set of potential arguments (potential argument nodes) of a predicate and extract features from them. The number of the candidate argument sets is exponential, thus we should consider only those corresponding to the most probable correct argument structures.

The usual approach (Toutanova et al., 2005) uses a traditional boundary classifier (TBC) to select the set of potential argument nodes. Such set can be associated with a subtree which in turn can be classified by means of a tree kernel function. This function intuitively measures to what extent a given candidate subtree is *compatible* with the subtree of a correct predicate argument structure. We can use it to define two different learning problems: (a) the simple classification of correct and incorrect predicate argument structures and (b) given the best m structures, we can train a re-ranker algorithm able to exploit argument inter-dependencies.

3.1 The Argument Spanning Trees (ASTs)

We consider predicate argument structures annotated in PropBank along with the corresponding TreeBank data as our object space. Given the target predicate node p and a node subset $s = \{n_1, .., n_k\}$ of the parse tree t, we define as the spanning tree root r the lowest common ancestor of $n_1, .., n_k$ and p. The node set spanning tree (NST) p_s is the subtree of t rooted in r from which the nodes that are neither ancestors nor descendants of any n_i or p are removed.

Since predicate arguments are associated with tree nodes (i.e. they exactly fit into syntactic constituents), we can define the *Argument Spanning Tree* (AST) of a predicate argument set, $\{p, \{a_1, .., a_n\}\}$, as the NST over such nodes, i.e. $p_{\{a_1,..,a_n\}}$. An AST corresponds to the *minimal* subtree whose leaves are all and only the words compounding the arguments and the predicate. For example, Figure 1 shows the parse tree of the sentence `"John took the book and read its title"`. $took_{\{Arg_0,Arg_1\}}$ and $read_{\{Arg_0,Arg_1\}}$ are two AST structures associated with the two predicates *took* and *read*, respectively. All the other possible subtrees, i.e. NSTs, are not valid ASTs for these two predicates. Note that classifying p_s in AST or NST for each node subset s of t is equivalent to solve the boundary detection problem.

The critical points for the AST classification are: (1) how to design suitable features for the characterization of valid structures. This requires a careful linguistic investigation about their significant properties. (2) How to deal with the exponential number of NSTs.

The first problem can be addressed by means of tree kernels over the ASTs. Tree kernel spaces are an alternative to the manual feature design as the learning machine, (e.g. SVMs) can select the most relevant features from a high dimensional space. In other words, we can use a tree kernel function to estimate the similarity between two ASTs (see Sec-

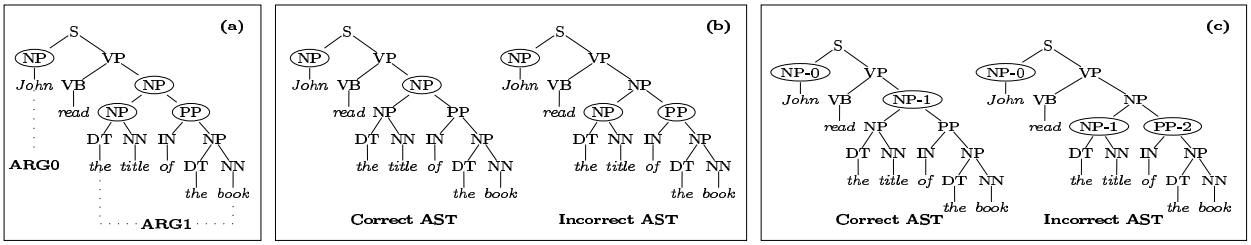


Figure 2: Two-step boundary classification. a) Sentence tree; b) Two candidate $ASTs$; c) Extended AST-Ord labeling

tion 2), hence avoiding to define explicit features.

The second problem can be approached in two ways:
(1) We can increase the recall of TBC to enlarge the set of candidate arguments. From such set, we can extract correct and incorrect argument structures. As the number of such structures will be rather small, we can apply the AST classifier to detect the correct ones.
(2) We can consider the classification probability provided by TBC and TRC (Pradhan et al., 2005a) and select the m most probable structures. Then, we can apply a re-ranking approach based on SVMs and tree kernels.

The re-ranking approach is the most promising one as suggested in (Toutanova et al., 2005) but it does not clearly reveal if tree kernels can be used to learn the difference between correct or incorrect argument structures. Thus it is interesting to study both the above approaches.

3.2 NST Classification

As we cannot classify all possible candidate argument structures, we apply the AST classifier just to detect the correct structures from a set of overlapping arguments. Given two nodes n_1 and n_2 of an NST, they overlap if either n_1 is ancestor of n_2 or vice versa. NSTs that contain overlapping nodes are not valid ASTs but subtrees of *NST*s may be valid $ASTs$. Assuming this, we define s as the set of potential argument nodes and we create two node sets $s_1 = s - \{n_1\}$ and $s_2 = s - \{n_2\}$. By classifying the two new NSTs p_{s_1} and p_{s_2} with the AST classifier, we can select the correct structures. Of course, this procedure can be generalized to a set of overlapping nodes greater than 2. However, considering that the Precision of TBC is generally high, the number of overlapping nodes is usually small.

Figure 2 shows a working example of the multi-stage classifier. In Frame (a), TBC labels as potential arguments (circled nodes) three overlapping nodes related to `Arg1`. This leads to two possible non-overlapping solutions (Frame (b)) but only the first one is correct. In fact, according to the second one the propositional phrase "*of the book*" would be incorrectly attached to the verbal predicate, i.e. in contrast with the parse tree. The AST classifier, applied to the two NSTs, is expected to detect this inconsistency and provide the correct output.

3.3 Re-ranking NSTs with Tree Kernels

To implement the re-ranking model, we follow the approach described in (Toutanova et al., 2005).

First, we use SVMs to implement the boundary TBC and role TRC *local* classifiers. As SVMs do not provide probabilistic output, we use the Platt's algorithm (Platt, 2000) and its revised version (Lin et al., 2003) to trasform scores into probabilities.

Second, we combine TBC and TRC probabilities to obtain the m most likely sequences s of tree nodes annotated with semantic roles. As argument constituents of the same verb cannot overlap, we generate sequences that respect such node constraint. We adopt the same algorithm described in (Toutanova et al., 2005). We start from the leaves and we select the m sequences that respect the constraints and at the same time have the highest joint probability of TBC and TRC.

Third, we extract the following feature representation:
(a) The ASTs associated with the predicate argument structures. To make faster the learning process and to try to only capture the most relevant features, we also experimented with a compact version of the

AST which is pruned at the level of argument nodes. (b) Attribute value features (standard features) related to the whole predicate structure. These include the features for each arguments (Gildea and Jurasfky, 2002) and global features like the sequence of argument labels, e.g. $\langle Arg0, Arg1, ArgM \rangle$.

Finally, we prepare the training examples for the re-ranker considering the m best annotations of each predicate structure. We use the approach adopted in (Shen et al., 2003), which generates all possible pairs from the m examples, i.e. $\binom{m}{2}$ pairs. Each pair is assigned to a positive example if the first member of the pair has a higher score than the second member. The score that we use is the F1 measure of the annotated structure with respect to the gold standard. More in detail, given training/testing examples $e_i = \langle t_i^1, t_i^2, v_i^1, v_i^2 \rangle$, where t_i^1 and t_i^2 are two ASTs and v_i^1 and v_i^2 are two feature vectors associated with two candidate predicate structures s_1 and s_2, we define the following kernels:

$$1) \quad K_{tr}(e_1, e_2) = K_t(t_1^1, t_2^1) + K_t(t_1^2, t_2^2) - K_t(t_1^1, t_2^2) - K_t(t_1^2, t_2^1),$$

where t_i^j is the j-th AST of the pair e_i, K_t is the tree kernel function defined in Section 2 and $i, j \in \{1, 2\}$.

$$2) \quad K_{pr}(e_1, e_2) = K_p(v_1^1, v_2^1) + K_p(v_1^2, v_2^2) - K_p(v_1^1, v_2^2) - K_p(v_1^2, v_2^1),$$

where v_i^j is the j-th feature vector of the pair e_i and K_p is the polynomial kernel applied to such vectors.

The final kernel that we use for re-ranking is the following:

$$K(e_1, e_2) = \frac{K_{tr}(e_1, e_2)}{|K_{tr}(e_1, e_2)|} + \frac{K_{pr}(e_1, e_2)}{|K_{pr}(e_1, e_2)|}$$

Regarding tree kernel feature engineering, the next section show how we can generate more effective features given an established kernel function.

3.4 Tree kernel feature engineering

Consider the Frame (b) of Figure 2, it shows two perfectly identical NSTs, consequently, their fragments will also be equal. This prevents the algorithm to learn something from such examples. To solve the problem, we can enrich the $NSTs$ by marking their argument nodes with a progressive number, starting from the leftmost argument. For example, in the first NST of Frame (c), we mark as `NP-0` and `NP-1` the first and second argument nodes whereas in the second NST we trasform the three argument node labels in `NP-0`, `NP-1` and `PP-2`. We will refer to the resulting structure as a AST-Ord (ordinal number). This simple modification allows the tree kernel to generate different argument structures for the above NSTs. For example, from the first NST in Figure 2.c, the fragments `[NP-1 [NP][PP]]`, `[NP [DT][NN]]` and `[PP [IN][NP]]` are generated. They do not match anymore with the `[NP-0 [NP][PP]]`, `[NP-1 [DT][NN]]` and `[PP-2 [IN][NP]]` fragments generated from the second NST in Figure 2.c.

Additionally, it should be noted that the semantic information provided by the role type can remarkably help the detection of correct or incorrect predicate argument structures. Thus, we can enrich the argument node label with the role type, e.g. the `NP-0` and `NP-1` of the correct AST of Figure 2.c become `NP-Arg0` and `NP-Arg1` (not shown in the figure). We refer to this structure as AST-Arg. Of course, to apply the AST-Arg classifier, we need that TRC labels the arguments detected by TBC.

4 The experiments

The experiments were carried out within the setting defined in the CoNLL-2005 Shared Task (Carreras and Màrquez, 2005). In particular, we adopted the Charniak parse trees available at `www.lsi.upc.edu/~srlconll/` along with the official performance evaluator.

All the experiments were performed with the SVM-light-TK software available at `http://ai-nlp.info.uniroma2.it/moschitti/` which encodes ST and SST kernels in SVM-light (Joachims, 1999). For TBC and TRC, we used the linear kernel with a regularization parameter (option `-c`) equal to 1. A cost factor (option `-j`) of 10 was adopted for TBC to have a higher Recall, whereas for TRC, the cost factor was parameterized according to the maximal accuracy of each argument class on the validation set. For the AST-based classifiers we used a λ equal to 0.4 (see (Moschitti, 2004)).

	Section 21			Section 23		
AST Class.	P.	R.	F_1	P.	R.	F_1
–	69.8	77.9	73.7	62.2	77.1	68.9
Ord	73.7	81.2	77.3	63.7	80.6	71.2
Arg	73.6	84.7	78.7	64.2	82.3	72.1

Table 1: AST, AST-Ord, and AST-Arg performance on sections 21 and 23.

4.1 Classification of whole predicate argument structures

In these experiments, we trained TBC on sections 02-08 whereas, to achieve a very accurate role classifier, we trained TRC on all sections 02-21. To train the AST, AST-Ord (AST with ordinal numbers in the argument nodes), and AST-Arg (AST with argument type in the argument nodes) classifiers, we applied the TBC and TRC over sections 09-20. Then, we considered all the structures whose automatic annotation showed at least an argument overlap. From these, we extracted 30,220 valid ASTs and 28,143 non-valid ASTs, for a total of 183,642 arguments.

First, we evaluate the accuracy of the AST-based classifiers by extracting 1,975 ASTs and 2,220 non-ASTs from Section 21 and the 2,159 ASTs and 3,461 non-ASTs from Section 23. The accuracy derived on Section 21 is an upperbound for our classifiers since it is obtained using an ideal syntactic parser (the Charniak's parser was trained also on Section 21) and an ideal role classifier.

Table 1 shows Precision, Recall and F_1 measures of the AST-based classifiers over the above NSTs. Rows 2, 3 and 4 report the performance of AST, AST-Ord, and AST-Arg classifiers, respectively. We note that: (a) The impact of parsing accuracy is shown by the gap of about 6% points between sections 21 and 23. (b) The ordinal numbering of arguments (Ord) and the role type information (Arg) provide tree kernels with more meaningful fragments since they improve the basic model of about 4%. (c) The deeper semantic information generated by the Arg labels provides useful clues to select correct predicate argument structures since it improves the Ord model on both sections.

Second, we measured the impact of the AST-based classifiers on the accuracy of both phases of semantic role labeling. Table 2 reports the results on sections 21 and 23. For each of them, Precision, Recall and F_1 of different approaches to boundary identification (bnd) and to the complete task, i.e. boundary and role classification (bnd+class) are shown. Such approaches are based on different strategies to remove the overlaps, i.e. with the AST, AST-Ord and AST-Arg classifiers and using the baseline (RND), i.e. a random selection of non-overlapping structures. The baseline corresponds to the system based on TBC and TRC[1].

We note that: (a) for any model, the boundary detection F_1 on Section 21 is about 10 points higher than the F_1 on Section 23 (e.g. 87.0% vs. 77.9% for RND). As expected the parse tree quality is very important to detect argument boundaries. (b) On the real test (Section 23) the classification introduces labeling errors which decrease the accuracy of about 5% (77.9 vs 72.9 for RND). (c) The Ord and Arg approaches constantly improve the baseline F_1 of about 1%. Such poor impact does not surprise as the overlapping structures are a small percentage of the test set, thus the overall improvement cannot be very high.

Third, the comparison with the CoNLL 2005 results (Carreras and Màrquez, 2005) can only be carried out with respect to the whole SRL task (bnd+class in table 2) since boundary detection versus role classification is generally not provided in CoNLL 2005. Moreover, our best global result, i.e. 73.9%, was obtained under two severe experimental factors: a) the use of just 1/3 of the available training set, and b) the usage of the linear SVM model for the TBC classifier, which is much faster than the polynomial SVMs but also less accurate. However, we note the promising results of the AST meta-classifier, which can be used with any of the best figure CoNLL systems.

Finally, the overall results suggest that the tree kernel model is robust to parse tree errors since preserves the same improvement across trees derived with different accuracy, i.e. the *semi-automatic* trees of Section 21 and the automatic tree of Section 23. Moreover, it shows a high accuracy for the classification of correct and incorrect ASTs. This last property is quite interesting as the best SRL systems

[1]We needed to remove the overlaps from the baseline outcome in order to apply the CoNLL evaluator.

(Punyakanok et al., 2005; Toutanova et al., 2005; Pradhan et al., 2005b) were obtained by exploiting the information on the whole predicate argument structure.

Next section shows our preliminary experiments on re-ranking using the AST kernel based approach.

4.2 Re-ranking based on Tree Kernels

In these experiments, we used the output of TBC and TRC[2] to provide an SVM tree kernel with a ranked list of predicate argument structures. More in detail, we applied a Viterbi-like algorithm to generate the 20 most likely annotations for each predicate structure, according to the joint probabilistic model of TBC and TRC. We sorted such structures based on their F_1 measure and used them to learn the SVM re-ranker described in 3.3.

For training, we used Sections 12, 14, 15, 16 and 24, which contain 24,729 predicate structures. For each of them, we considered the 5 annotations having the highest F1 score (i.e. 123,674 NSTs) on the span of the 20 best annotations provided by Viterbi algorithm. With such structures, we obtained 294,296 pairs used to train the SVM-based re-ranker. As the number of such structures is very large the SVM training time was very high. Thus, we sped up the learning process by using only the ASTs associated with the core arguments. From the test sentences (which contain 5,267 structures), we extracted the 20 best Viterbi annotated structures, i.e. 102,343 (for a total of 315.531 pairs), which were used for the following experiments:

First, we selected the best annotation (according to the F_1 provided by the gold standard annotations) out of the 20 provided by the Viterbi's algorithm. The resulting F_1 of 88.59% is the upperbound of our approach.

Second, we selected the top ranked annotation indicated by the Viterbi's algorithm. This provides our baseline F_1 measure, i.e. 75.91%. Such outcome is slightly higher than our official CoNLL result (Moschitti et al., 2005) obtained without converting SVM scores into probabilities.

Third, we applied the SVM re-ranker to select the best structures according to the core roles. We achieved 80.68% which is practically equal to the result obtained in (Punyakanok et al., 2005; Carreras and Màrquez, 2005) for core roles, i.e. 81%. Their overall F1 which includes all the arguments was 79.44%. This confirms that the classification of the non-core roles is more complex than the other arguments.

Finally, the high computation time of the re-ranker prevented us to use the larger structures which include all arguments. The major complexity issue was the slow training and classification time of SVMs. The time needed for tree kernel function was not so problematic as we could use the fast evaluation proposed in (Moschitti, 2006). This roughly reduces the computation time to the one required by a polynomial kernel. The real burden is therefore the learning time of SVMs that is quadratic in the number of training instances. For example, to carry out the re-ranking experiments required approximately one month of a 64 bits machine (2.4 GHz and 4Gb Ram). To solve this problem, we are going to study the impact on the accuracy of fast learning algorithms such as the Voted Perceptron.

5 Related Work

Recently, many kernels for natural language applications have been designed. In what follows, we highlight their difference and properties.

The tree kernel used in this article was proposed in (Collins and Duffy, 2002) for syntactic parsing re-ranking. It was experimented with the Voted Perceptron and was shown to improve the syntactic parsing. In (Cumby and Roth, 2003), a feature description language was used to extract structural features from the syntactic shallow parse trees associated with named entities. The experiments on the named entity categorization showed that when the description language selects an adequate set of tree fragments the Voted Perceptron algorithm increases its classification accuracy. The explanation was that the complete tree fragment set contains many irrelevant features and may cause overfitting. In (Punyakanok et al., 2005), a set of different syntactic parse trees, e.g. the n best trees generated by the Charniak's parser, were used to improve the SRL accuracy. These different sources of syntactic information were used to generate a set of different SRL

[2]With the aim of improving the state-of-the-art, we applied the polynomial kernel for all basic classifiers, at this time. We used the models developed during our participation to the CoNLL 2005 shared task (Moschitti et al., 2005).

	Section 21								Section 23							
	bnd				bnd+class				bnd				bnd+class			
	AST Classifier			RND	AST Classifier			RND	AST Classifier			RND	AST Classifier			RND
	-	Ord	Arg		-	Ord	Arg		-	Ord	Arg		-	Ord	Arg	
P.	87.5	88.3	88.3	86.9	85.5	86.3	86.4	85.0	78.6	79.0	79.3	77.8	73.1	73.5	73.4	72.3
R.	87.3	88.1	88.3	87.1	85.7	86.5	86.8	85.6	78.1	78.4	78.7	77.9	73.8	74.1	74.4	73.6
F_1	87.4	88.2	88.3	87.0	85.6	86.4	86.6	85.3	78.3	78.7	79.0	77.9	73.4	73.8	73.9	72.9

Table 2: Semantic Role Labeling performance on automatic trees using AST-based classifiers.

outputs. A joint inference stage was applied to resolve the inconsistency of the different outputs. In (Toutanova et al., 2005), it was observed that there are strong dependencies among the labels of the semantic argument nodes of a verb. Thus, to approach the problem, a re-ranking method of role sequences labeled by a TRC is applied. In (Pradhan et al., 2005b), some experiments were conducted on SRL systems trained using different syntactic views.

6 Conclusions

Recent work on Semantic Role Labeling has shown that to achieve high labeling accuracy a joint inference on the whole predicate argument structure should be applied. As feature design for such task is complex, we can take advantage from kernel methods to model our intuitive knowledge about the n-ary predicate argument relations.

In this paper we have shown that we can exploit the properties of tree kernels to engineer syntactic features for the semantic role labeling task. The experiments suggest that (1) the information related to the whole predicate argument structure is important as it can improve the state-of-the-art and (2) tree kernels can be used in a joint model to generate relevant syntactic/semantic features. The real drawback is the computational complexity of working with SVMs, thus the design of fast algorithm is an interesting future work.

Acknowledgments

This research is partially supported by the PrestoSpace EU Project#: FP6-507336.

References

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL05*.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL02*.

Chad Cumby and Dan Roth. 2003. Kernel methods for relational learning. In *Proceedings of ICML03*, Washington, DC, USA.

Daniel Gildea and Daniel Jurasfky. 2002. Automatic labeling of semantic roles. *Computational Linguistic*, 28(3):496–530.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of LREC'02)*, Las Palmas, Spain.

H.T. Lin, C.J. Lin, and R.C. Weng. 2003. A note on platt's probabilistic outputs for support vector machines. Technical report, National Taiwan University.

Alessandro Moschitti, Bonaventura Coppola, Daniele Pighin, and Roberto Basili. 2005. Hierarchical semantic role labeling. In *Proceedings of CoNLL05 shared task*, Ann Arbor (MI), USA.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of ACL'04*, Barcelona, Spain.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning. In *Proceedings of EACL'06*, Trento, Italy.

J. Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. MIT Press.

Sameer Pradhan, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005a. Support vector learning for semantic argument classification. *Machine Learning Journal*.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005b. Semantic role labeling using different syntactic views. In *Proceedings ACL'05*.

V. Punyakanok, D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of IJCAI 2005*.

Libin Shen, Anoop Sarkar, and Aravind Joshi. 2003. Using ltag based features in parse reranking. In *Conference on EMNLP03*, Sapporo, Japan.

Kristina Toutanova, Penka Markova, and Christopher D. Manning. 2004. The leaf projection path view of parse trees: Exploring string kernels for hpsg parse selection. In *In Proceedings of EMNLP04*.

Kristina Toutanova, Aria Haghighi, and Christopher Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL05*.

Can Human Verb Associations Help Identify Salient Features for Semantic Verb Classification?

Sabine Schulte im Walde
Computational Linguistics
Saarland University
Saarbrücken, Germany
schulte@coli.uni-sb.de

Abstract

This paper investigates whether human associations to verbs as collected in a web experiment can help us to identify salient verb features for semantic verb classes. Assuming that the associations model aspects of verb meaning, we apply a clustering to the verbs, as based on the associations, and validate the resulting verb classes against standard approaches to semantic verb classes, i.e. GermaNet and FrameNet. Then, various clusterings of the same verbs are performed on the basis of standard corpus-based types, and evaluated against the association-based clustering as well as GermaNet and FrameNet classes. We hypothesise that the corpus-based clusterings are better if the instantiations of the feature types show more overlap with the verb associations, and that the associations therefore help to identify salient feature types.

1 Introduction

There are a variety of manual semantic verb classifications; major frameworks are the Levin classes (Levin, 1993), *WordNet* (Fellbaum, 1998), and *FrameNet* (Fontenelle, 2003). The different frameworks depend on different instantiations of semantic similarity, e.g. Levin relies on verb similarity referring to syntax-semantic alternation behaviour, WordNet uses synonymy, and FrameNet relies on situation-based agreement as defined in Fillmore's frame semantics (Fillmore, 1982). As an alternative to the resource-intensive manual classifications, automatic methods such as classification and clustering are applied to induce verb classes from corpus data, e.g. (Merlo and Stevenson, 2001; Joanis and Stevenson, 2003; Korhonen et al., 2003; Stevenson and Joanis, 2003; Schulte im Walde, 2003; Ferrer, 2004). Depending on the types of verb classes to be induced, the automatic approaches vary their choice of verbs and classification/clustering algorithm. However, another central parameter for the automatic induction of semantic verb classes is the selection of verb features.

Since the target classification determines the similarity and dissimilarity of the verbs, the verb feature selection should model the similarity of interest. For example, Merlo and Stevenson (2001) classify 60 English verbs which alternate between an intransitive and a transitive usage, and assign them to three verb classes, according to the semantic role assignment in the frames; their verb features are chosen such that they model the syntactic frame alternation proportions and also heuristics for semantic role assignment. In larger-scale classifications such as (Korhonen et al., 2003; Stevenson and Joanis, 2003; Schulte im Walde, 2003), which model verb classes with similarity at the syntax-semantics interface, it is not clear which features are the most salient. The verb features need to relate to a behavioural component (modelling the syntax-semantics interplay), but the set of features which potentially influence the behaviour is large, ranging from structural syntactic descriptions and argument role fillers to adverbial adjuncts. In addition, it is not clear how fine-grained the features should be; for example, how much information is covered by low-level window co-occurrence vs. higher-order syntactic frame fillers?

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 69–76, New York City, June 2006.

In this paper, we investigate whether human associations to verbs can help us to identify salient verb features for semantic verb classes. We collected associations to German verbs in a web experiment, and hope that these associations represent a useful basis for a theory-independent semantic classification of the German verbs, assuming that the associations model a non-restricted set of salient verb meaning aspects. In a preparatory step, we perform an unsupervised clustering on the experiment verbs, as based on the verb associations. We validate the resulting verb classes (henceforth: *assoc-classes*) by demonstrating that they show considerable overlap with standard approaches to semantic verb classes, i.e. GermaNet and FrameNet. In the main body of this work, we compare the associations underlying the assoc-classes with standard corpus-based feature types: We check on how many of the associations we find among the corpus-based features, such as adverbs, direct object nouns, etc.; we hypothesise that the more associations are found as instantiations in a feature set, the better is a clustering as based on that feature type. We assess our hypothesis by applying various corpus-based feature types to the experiment verbs, and comparing the resulting classes (henceforth: *corpus-classes*) against the assoc-classes. On the basis of the comparison we intend to answer the question whether *the human associations help identify salient features to induce semantic verb classes,* i.e. do the corpus-based feature types which are identified on the basis of the associations outperform previous clustering results? By applying the feature choices to GermaNet and FrameNet, we address the question whether *the same types of features are salient for different types of semantic verb classes?*

In what follows, the paper presents the association data in Section 2 and the association-based classes in Section 3. In Section 4, we compare the associations with corpus-based feature types, and in Section 5 we apply the insights to induce semantic verb classes.

2 Verb Association Data

We obtained human associations to German verbs from native speakers in a web experiment (Schulte im Walde and Melinger, 2005). 330 verbs were selected for the experiment (henceforth: *experiment verbs*), from different semantic categories, and different corpus frequency bands. Participants were given 55 verbs each, and had 30 seconds per verb to type as many associations as they could. 299 native German speakers participated in the experiment, between 44 and 54 for each verb. In total, we collected 81,373 associations from 16,445 trials; each trial elicited an average of 5.16 responses with a range of 0-16.

All data sets were pre-processed in the following way: For each target verb, we quantified over all responses in the experiment. Table 1 lists the 10 most frequent response types for the verb *klagen* 'complain, moan, sue'. The responses were not distinguished according to polysemic senses of the verbs.

klagen 'complain, moan, sue'		
Gericht	'court'	19
jammern	'moan'	18
weinen	'cry'	13
Anwalt	'lawyer'	11
Richter	'judge'	9
Klage	'complaint'	7
Leid	'suffering'	6
Trauer	'mourning'	6
Klagemauer	'Wailing Wall'	5
laut	'noisy'	5

Table 1: Association frequencies for target verb.

In the clustering experiments to follow, the verb associations are considered as verb features. The underlying assumption is that verbs which are semantically similar tend to have similar associations, and are therefore assigned to common classes. Table 2 illustrates the overlap of associations for the polysemous *klagen* with a near-synonym of one of its senses, *jammern* 'moan'. The table lists those associations which were given at least twice for each verb; the total overlap was 35 association types.

klagen/jammern 'moan'		
Frauen	'women'	2/3
Leid	'suffering'	6/3
Schmerz	'pain'	3/7
Trauer	'mourning'	6/2
bedauern	'regret'	2/2
beklagen	'bemoan'	4/3
heulen	'cry'	2/3
nervig	'annoying'	2/2
nölen	'moan'	2/3
traurig	'sad'	2/5
weinen	'cry'	13/9

Table 2: Association overlap for target verbs.

3 Association-based Verb Classes

We performed a standard clustering on the 330 experiment target verbs: The verbs and their features were taken as input to agglomerative (bottom-up) hierarchical clustering. As similarity measure in the clustering procedure (i.e. to determine the distance/similarity for two verbs), we used the *skew divergence*, a smoothed variant of the *Kullback-Leibler divergence* (Lee, 2001). The goal of these experiments was not to explore the optimal feature combination; thus, we rely on previous experiments and parameter settings, cf. Schulte im Walde (2003).

Our claim is that the hierarchical verb classes and their underlying features (i.e. the verb associations) represent a useful basis for a theory-independent semantic classification of the German verbs. To support this claim, we validated the assoc-classes against standard approaches to semantic verb classes, i.e. *GermaNet* as the German WordNet (Kunze, 2000), and the German counterpart of FrameNet in the *Salsa* project (Erk et al., 2003). Details of the validation can be found in (Schulte im Walde, 2006); the main issues are as follows.

We did not directly compare the assoc-classes against the GermaNet/FrameNet classes, since not all of our 330 experiments verbs were covered by the two resources. Instead, we replicated the above cluster experiment for a reduced number of verbs: We extracted those classes from the resources which contain association verbs; light verbs, non-association verbs, other classes as well as singletons were disregarded. This left us with 33 classes from GermaNet, and 38 classes from FrameNet. These remaining classifications are polysemous: The 33 GermaNet classes contain 71 verb senses which distribute over 56 verbs, and the 38 FrameNet classes contain 145 verb senses which distribute over 91 verbs. Based on the 56/91 verbs in the two gold standard resources, we performed two cluster analyses, one for the GermaNet verbs, and one for the FrameNet verbs. As for the complete set of experiments verbs, we performed a hierarchical clustering on the respective subsets of the experiment verbs, with their associations as verb features. The actual validation procedure then used the reduced classifications: The resulting analyses were evaluated against the resource classes on each level in the hierarchies, i.e. from 56/91 classes to 1 class. As evaluation measure, we used a pair-wise measure which calculates precision, recall and a harmonic f-score as follows: Each verb pair in the cluster analysis was compared to the verb pairs in the gold standard classes, and evaluated as true or false positive (Hatzivassiloglou and McKeown, 1993).

The association-based clusters show overlap with the lexical resource classes of an f-score of 62.69% (for 32 verb classes) when comparing to GermaNet, and 34.68% (for 10 verb classes) when comparing to FrameNet. The corresponding upper bounds are 82.35% for GermaNet and 60.31% for FrameNet.[1] The comparison therefore demonstrates considerable overlap between association-based classes and existing semantic classes. The different results for the two resources are due to their semantic background (i.e. capturing synonymy vs. situation-based agreement), the numbers of verbs, and the degrees of ambiguity (an average of 1.6 senses per verb in FrameNet, as compared to 1.3 senses in GermaNet).

The purpose of the validation against semantic resources was to demonstrate that a clustering as based on the verb associations and a standard clustering setting compares well with existing semantic classes. We take the positive validation results as justification to use the assoc-classes as source for cluster information: The clustering defines the verbs in a common association-based class, and the features which are relevant for the respective class. For example, the 100-class analysis contains a class with the verbs *bedauern* 'regret', *heulen* 'cry', *jammern* 'moan', *klagen* 'complain, moan, sue', *verzweifeln* 'become desperate', and *weinen* 'cry', with the most distinctive features *Trauer* 'mourning', *weinen* 'cry', *traurig* 'sad', *Tränen* 'tears', *jammern* 'moan', *Angst* 'fear', *Mitleid* 'pity', *Schmerz* 'pain'.

4 Exploring Semantic Class Features

Our claim is that the features underlying the association-based classes help us guide the feature selection process in future clustering experiments, because we know which semantic classes are based

[1] The upper bounds are below 100%, because the hierarchical clustering assigns a verb to only one cluster, but the lexical resources contain polysemy. We created a hard version of the lexical resource classes where we randomly chose one sense of each polysemous verb, to calculate the upper bounds.

on which associations/features. We rely on the assoc-classes in the 100-class analysis of the hierarchical clustering[2] and features which exist for at least two verbs in a common class (and therefore hint to a minimum of verb similarity), and compare the associations underlying the assoc-classes with standard corpus-based feature types: We check on how many of the associations we find among the corpus-based features, such as adverbs, direct object nouns, etc. There are various possibilities to determine corpus-based features that potentially cover the associations; we decided in favour of feature types which have been suggested in related work:

a) Grammar-based relations: Previous work on distributional similarity has focused either on a specific word-word relation (such as Pereira et al. (1993) and Rooth et al. (1999) referring to a direct object noun for describing verbs), or used any syntactic relationship detected by a chunker or a parser (such as Lin (1998) and McCarthy et al. (2003)). We used a statistical grammar (Schulte im Walde, 2003) to filter all verb-noun pairs where the nouns represent nominal heads in NPs or PPs in syntactic relation to the verb (subject, object, adverbial function, etc.), and to filter all verb-adverb pairs where the adverbs modify the verbs.

b) Co-occurrence window: In previous work (Schulte im Walde and Melinger, 2005), we showed that only 28% of all noun associates were identified by the above statistical grammar as subcategorised nouns, but 69% were captured by a 20-word co-occurrence window in a 200-million word newspaper corpus. This finding suggests to use a co-occurrence window as alternative source for verb features, as compared to specific syntactic relations. We therefore determined the co-occurring words for all experiment verbs in a 20-word window (i.e. 20 words preceding and following the verb), irrespective of the part-of-speech of the co-occurring words.

Relying on the verb information extracted for a) and b), we checked for each verb-association pair whether it occurred among the grammar or window pairs. Table 3 illustrates which proportions of the associations we found in the two resource types. For the grammar-based relations, we checked argument NPs and PPs (as separate sets and together), and in addition we checked verb-noun pairs in the most common specific NP functions: *n* refers to the (nominative) intransitive subject, *na* to the transitive subject, and *ng* to the transitive (accusative) object. For the windows, *all* checks on co-occurrence of verbs and associations in the whole 200-million word corpus. *cut* also checks the whole corpus, but disregards the most and least frequent co-occurring words: verb-word pairs were only considered if the co-occurrence frequency of the word over all verbs was above 100 (disregarding low frequency pairs) and below 200,000 (disregarding high frequency pairs). Using the cut-offs, we can distinguish the relevance of high- and low-frequency features. Finally, *ADJ, ADV, N, V* perform co-occurrence checks for the whole corpus, but breaks down the *all* results with respect to the association part-of-speech.

As one would have expected, most of the associations (66%) were found in the 20-word co-occurrence window, because the window is neither restricted to a certain part-of-speech, nor to a certain grammar relation; in addition, the window is potentially larger than a sentence. Applying the frequency cut-offs reduces the overlap of association types and co-occurring words to 58%. Specifying the window results for the part-of-speech types illustrates that the nouns play the most important role in describing verb meaning (39% of the verb association types in the assoc-classes were found among the nouns in the corpus windows, 16% among the verbs, 9% among the adjectives, and 2% among the adverbs).[3]

The proportions of the nouns with a specific grammar relationship to the verbs show that we find more associations among direct objects than intransitive/transitive subjects. This insight confirms the assumption in previous work where only direct object nouns were used as salient features in distributional verb similarity, such as Pereira et al. (1993). However, the proportions are all below 10%. Considering all NPs and/or PPs, we find that the proportions increase for the NPs, and that the NPs play a more important role than the PPs. This insight confirms work on distributional similarity where not only direct object nouns, but all functional nouns

[2]The exact number of classes or the verb-per-class ratio are not relevant for investigating the use of associations.

[3]Caveat: These numbers correlate with the part-of-speech types of all associate responses: 62% of the responses were nouns, 25% verbs, 11% adjectives, and 2% adverbs.

Features	grammar relations						
	n	na	na	NP	PP	NP&PP	ADV
Cov. (%)	3.82	4.32	6.93	12.23	5.36	14.08	3.63

Features	co-occurrence: window-20					
	all	cut	ADJ	ADV	N	V
Cov. (%)	66.15	57.79	9.13	1.72	39.27	15.51

Table 3: Coverage of verb association features by grammar/window resources.

were considered as verb features, such as Lin (1998) and McCarthy et al. (2003). Of the adverb associations, we find only a small proportion among the parsed adverbs. All in all, the proportions of association types among the nouns/adverbs with a syntactic relationship to the verbs are rather low. Comparing the NP/PP proportions with the window noun proportions shows that salient verb features are not restricted to certain syntactic relationships, but also appear in a less restricted context window.

5 Inducing Verb Classes with Corpus-based Features

In the final step, we applied the corpus-based feature types to clusterings. The goal of this step was to determine whether the feature exploration helped to identify salient verb features, and whether we can outperform previous clustering results. The clustering experiments were as follows: The 330 experiment verbs were instantiated by the feature types we explored in Section 4. As for the assoc-classes, we then performed an agglomerative hierarchical clustering. We cut the hierarchy at a level of 100 clusters, and evaluated the clustering against the 100-class analysis of the original assoc-classes. We expect that feature types with a stronger overlap with the association types result in a better clustering result. The assumption is that the associations are salient feature for verb clustering, and the better we model the associations with grammar-based or window-based features, the better the clustering.

For checking the clusterings with respect to the semantic class type, we also applied the corpus-based features to GermaNet and FrameNet classes.

- **GermaNet:** We randomly extracted 100 verb classes from all GermaNet synsets, and created a hard classification for these classes, by randomly deleting additional senses of a verb so as to leave only one sense for each verb. This selection made the GermaNet classes comparable to the assoc-classes in size and polysemy. The 100 classes contain 233 verbs. Again, we performed an agglomerative hierarchical clustering on the verbs (as modelled by the different feature types). We cut the hierarchy at a level of 100 clusters, which corresponds to the number of GermaNet classes, and evaluated against the GermaNet classes.

- **FrameNet:** In a pre-release version from May 2005, there were 484 verbs in 214 German FrameNet classes. We disregarded the high-frequency verbs *gehen, geben, sehen, kommen, bringen* which were assigned to classes mostly on the basis of multi-word expressions they are part of. In addition, we disregarded two large classes which contained mostly support verbs, and we disregarded singletons. Finally, we created a hard classification of the classes, by randomly deleting additional senses of a verb so as to leave only one sense for each verb. The classification then contained 77 classes with 406 verbs. Again, we performed an agglomerative hierarchical clustering on the verbs (as modelled by the different feature types). We cut the hierarchy at a level of 77 clusters, which corresponds to the number of FrameNet classes, and evaluated against the FrameNet classes.

For the evaluation of the clustering results, we calculated the *accuracy* of the clusters, a cluster similarity measure that has been applied before, cf. (Stevenson and Joanis, 2003; Korhonen et al., 2003).[4] Accuracy is determined in two steps:

[4]Note that we can use *accuracy* for the evaluation because we have a fixed cut in the hierarchy as based on the gold standard, as opposed to the evaluation in Section 3 where we explored the optimal cut level.

	frames		grammar relations						
	f-pp	f-pp-pref	n	na	na	NP	PP	NP&PP	ADV
Assoc	37.50	37.80	35.90	37.18	39.25	39.14	37.97	**41.28**	38.53
GN	46.98	49.14	**58.01**	53.37	51.90	53.10	54.21	51.77	51.82
FN	33.50	32.76	29.46	30.13	32.74	34.16	28.72	33.91	**35.24**

	co-occurrence: window-20					
	all	cut	ADJ	ADV	N	V
Assoc	39.33	**39.45**	37.31	36.89	39.33	38.84
GN	51.53	52.42	50.88	47.79	**52.86**	49.12
FN	missing	32.84	31.08	31.00	**34.24**	31.75

Table 4: Accuracy for induced verb classes.

1. For each class in the cluster analysis, the gold standard class with the largest intersection of verbs is determined. The number of verbs in the intersection ranges from one verb only (in case all clustered verbs are in different classes in the gold standard) to the total number of verbs in a cluster (in case all clustered verbs are in the same gold standard class).
2. Accuracy is calculated as the proportion of the verbs in the clusters covered by the same gold standard classes, divided by the total number of verbs in the clusters. The upper bound of the accuracy measure is 1.

Table 4 shows the accuracy results for the three types of classifications (assoc-classes, GermaNet, FrameNet), and the grammar-based and window-based features. We added frame-based features, as to compare with earlier work: The frame-based features provide a feature description over 183 syntactic frame types including PP type specification (f-pp), and the same information plus coarse selectional preferences for selected frame slots, as obtained from GermaNet top-level synsets (f-pp-pref), cf. (Schulte im Walde, 2003). The following questions are addressed with respect to the result table.

1. Do the results of the clusterings with respect to the underlying feature types correspond to the overlap of associations and feature types, cf. Table 3?
2. Do the corpus-based feature types which were identified on the basis of the associations outperform previous clustering results?
3. Do the results generalise over the semantic class type?

First of all, there is no correlation between the overlap of associations and feature types on the one hand and the clustering results as based on the feature types on the other hand (Pearson's correlation, $p>.1$), neither for the assoc-classes or the GermaNet or FrameNet classes. The human associations therefore did not contribute to identify salient feature types, as we had hoped. In some specific cases, we find corresponding patterns; for example, the clustering results for the intransitive and transitive subject and the transitive object correspond to the overlap values for the assoc-classes and FrameNet: n $<$ na $<$ na. Interestingly, the GermaNet clusterings behave in the opposite direction.

Comparing the grammar-based relations with each other shows that for the assoc-classes using all NPs is better than restricting the NPs to (subject) functions, and using both NPs and PPs is best; similarly for the FrameNet classes where using all NPs is the second best results (but adverbs). Differently, for the GermaNet classes the specific function of intransitive subjects outperforms the more general feature types, and the PPs are still better than the NPs. We conclude that not only there is no correlation between the association overlap and feature types, but in addition the most successful feature types vary strongly with respect to the gold standard. None of the differences within the feature groups (n/na/na and NP/PP/NP&PP) are significant ($\chi^2, df = 1, \alpha = 0.05$). The adverbial features are surprisingly successful in all three clusterings, in some cases outperforming the noun-based features.

Comparing the grammar-based clustering results with previous results, the grammar-based features outperform the frame-based features in all clusterings for the GermaNet verbs. For the FrameNet

verbs and the experiment verbs, they outperform the frame-based features only in specific cases. The adverbial features outperform the frame-based features in any clustering. However, none of the differences between the frame-based clusterings and the grammar-based clusterings are significant ($\chi^2, df = 1, \alpha = 0.05$).

For all gold standards, the best window-based clustering results are below the best grammar-based results. Especially the *all* results demonstrate once more the missing correlation between association/feature overlap and clustering results. However, it is interesting that the clusterings based on window co-occurrence are not significantly worse (and in some cases even better) than the clusterings based on selected grammar-based functions. This means that a careful choice and extraction of specific relationships for verb features does not have a significant impact on semantic classes.

Comparing the window-based features against each other shows that even though we discovered a much larger proportion of association types in an unrestricted window *all* than elsewhere, the results in the clusterings do not differ accordingly. Applying the frequency cut-offs has almost no impact on the clustering results, which means that it does no harm to leave away the rather unpredictable features. Somehow expected but nevertheless impressive is the fact that only considering nouns as co-occurring words is as successful as considering all words independent of the part-of-speech.

Finally, the overall accuracy values are much better for the GermaNet clusterings than for the experiment-based and the FrameNet clusterings. The differences are all significant ($\chi^2, df = 1, \alpha = 0.05$). The reason for these large differences could be either (a) that the clustering task was easier for the GermaNet verbs, or (b) that the differences are caused by the underlying semantics. We argue against case (a) since we deliberately chose the same number of classes (100) as for the association-based gold standard; however, the verbs-per-class ratio for GermaNet vs. the assoc-classes and the FrameNet classes is different (2.33 vs. 3.30/5.27) and we cannot be sure about this influence. In addition, the average verb frequencies in the GermaNet classes (calculated in a 35 million word newspaper corpus) are clearly below those in the other two classifications (1,040 as compared to 2,465 and 1,876), and there are more low-frequency verbs (98 out of 233 verbs (42%) have a corpus frequency below 50, as compared to 41 out of 330 (12%) and 54 out of 406 (13%)). In the case of (b), the difference in the semantic class types is modelling synonyms with GermaNet as opposed to situation-based agreement in FrameNet. The association-based class semantics is similar to FrameNet, because the associations are unrestricted in their semantic relation to the experiment verb (Schulte im Walde and Melinger, 2005).

6 Summary

The questions we posed in the beginning of this paper were (i) whether human associations help identify salient features to induce semantic verb classes, and (ii) whether the same types of features are salient for different types of semantic verb classes. An association-based clustering with 100 classes served as source for identifying a set of potentially salient verb features, and a comparison with standard corpus-based features determined proportions of feature overlap. Applying the standard feature choices to verbs underlying three gold standard verb classifications showed that (a) in our experiments there is no correlation between the overlap of associations and feature types and the respective clustering results. The associations therefore did not help in the specific choice of corpus-based features, as we had hoped. However, the assumption that window-based features do contribute to semantic verb classes – this assumption came out of an analysis of the associations – was confirmed: simple window-based features were not significantly worse (and in some cases even better) than selected grammar-based functions. This finding is interesting because window-based features have often been considered too simple for semantic similarity, as opposed to syntax-based features. (b) Several of the grammar-based nominal and adverbial features and also the window-based features outperformed feature sets in previous work, where frame-based features (plus prepositional phrases and coarse selectional preference information) were used. Surprisingly well did adverbs: they only represent a small number of verb features, but obviously this small selection can outperform frame-based features and even some nomi-

nal features. (c) The clustering results were significantly better for the GermaNet clusterings than for the experiment-based and the FrameNet clusterings, so the chosen feature sets might be more appropriate for the synonymy-based than the situation-based classifications.

Acknowledgements Thanks to Christoph Clodo and Marty Mayberry for their system administrative help when running the cluster analyses.

References

Katrin Erk, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2003. Towards a Resource for Lexical Semantics: A Large German Corpus with Extensive Semantic Annotation. In *Proceedings of the 41st Annual Metting of the Association for Computational Linguistics*, Sapporo, Japan.

Christiane Fellbaum, editor. 1998. *WordNet – An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA.

Eva Esteve Ferrer. 2004. Towards a Semantic Classification of Spanish Verbs based on Subcategorisation Information. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain.

Charles J. Fillmore. 1982. Frame Semantics. *Linguistics in the Morning Calm*, pages 111–137.

Thierry Fontenelle, editor. 2003. *FrameNet and Frame Semantics*, volume 16(3) of *International Journal of Lexicography*. Oxford University Press. Special issue devoted to FrameNet.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1993. Towards the Automatic Identification of Adjectival Scales: Clustering Adjectives According to Meaning. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 172–182, Columbus, Ohio.

Eric Joanis and Suzanne Stevenson. 2003. A General Feature Space for Automatic Verb Classification. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary.

Anna Korhonen, Yuval Krymolowski, and Zvika Marx. 2003. Clustering Polysemic Subcategorization Frame Distributions Semantically. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 64–71, Sapporo, Japan.

Claudia Kunze. 2000. Extension and Use of GermaNet, a Lexical-Semantic Database. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, pages 999–1002, Athens, Greece.

Lillian Lee. 2001. On the Effectiveness of the Skew Divergence for Statistical Language Analysis. *Artificial Intelligence and Statistics*, pages 65–72.

Dekang Lin. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of the 17th International Conference on Computational Linguistics*, Montreal, Canada.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a Continuum of Compositionality in Phrasal Verbs. In *Proceedings of the ACL-SIGLEX Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, Sapporo, Japan.

Paola Merlo and Suzanne Stevenson. 2001. Automatic Verb Classification Based on Statistical Distributions of Argument Structure. *Computational Linguistics*, 27(3):373–408.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional Clustering of English Words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a Semantically Annotated Lexicon via EM-Based Clustering. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, Maryland, MD.

Sabine Schulte im Walde and Alissa Melinger. 2005. Identifying Semantic Relations and Functional Properties of Human Verb Associations. In *Proceedings of the joint Conference on Human Language Technology and Empirial Methods in Natural Language Processing*, pages 612–619, Vancouver, Canada.

Sabine Schulte im Walde. 2003. *Experiments on the Automatic Induction of German Semantic Verb Classes*. Ph.D. thesis, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart. Published as AIMS Report 9(2).

Sabine Schulte im Walde. 2006. Human Verb Associations as the Basis for Gold Standard Verb Classes: Validation against GermaNet and FrameNet. In *Proceedings of the 5th Conference on Language Resources and Evaluation*, Genoa, Italy.

Suzanne Stevenson and Eric Joanis. 2003. Semi-supervised Verb Class Discovery Using Noisy Features. In *Proceedings of the 7th Conference on Natural Language Learning*, pages 71–78, Edmonton, Canada.

Applying Alternating Structure Optimization to Word Sense Disambiguation

Rie Kubota Ando
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, U.S.A.
rie1@us.ibm.com

Abstract

This paper presents a new application of the recently proposed machine learning method *Alternating Structure Optimization (ASO)*, to word sense disambiguation (WSD). Given a set of WSD problems and their respective labeled examples, we seek to improve overall performance on that set by using all the labeled examples (irrespective of target words) for the entire set in learning a disambiguator for each individual problem. Thus, in effect, on each individual problem (e.g., disambiguation of "art") we benefit from training examples for other problems (e.g., disambiguation of "bar", "canal", and so forth). We empirically study the effective use of ASO for this purpose in the multi-task and semi-supervised learning configurations. Our performance results rival or exceed those of the previous best systems on several Senseval lexical sample task data sets.

1 Introduction

Word sense disambiguation (WSD) is the task of assigning pre-defined senses to words occurring in some context. An example is to disambiguate an occurrence of "bank" between the "money bank" sense and the "river bank" sense. Previous studies e.g., (Lee and Ng, 2002; Florian and Yarowsky, 2002), have applied supervised learning techniques to WSD with success.

A practical issue that arises in supervised WSD is the paucity of labeled examples (sense-annotated data) available for training. For example, the training set of the Senseval-2[1] English lexical sample task has only 10 labeled training examples per sense on average, which is in contrast to nearly 6K training examples per name class (on average) used for the CoNLL-2003 named entity chunking shared task[2]. One problem is that there are so many words and so many senses that it is hard to make available a sufficient number of labeled training examples for each of a large number of target words.

On the other hand, this indicates that the total number of available labeled examples (irrespective of target words) can be relatively large. A natural question to ask is whether we can effectively use *all* the labeled examples (irrespective of target words) for learning on each individual WSD problem.

Based on these observations, we study a new application of *Alternating Structure Optimization (ASO)* (Ando and Zhang, 2005a; Ando and Zhang, 2005b) to WSD. ASO is a recently proposed machine learning method for learning predictive structure (i.e., information useful for predictions) shared by multiple prediction problems via joint empirical risk minimization. It has been shown that on several tasks, performance can be significantly improved by a semi-supervised application of ASO, which obtains useful information from *unlabeled data* by learning automatically created prediction problems. In addition to such semi-supervised learning, this paper explores ASO *multi-task learning*, which learns a number of WSD problems simultaneously to exploit the inherent predictive structure shared by these WSD problems. Thus, in effect, each individual problem (e.g., disambiguation of "art") benefits from *labeled training examples for other problems* (e.g., disambiguation of "bar", disambiguation of "canal", and so forth).

The notion of benefiting from training data for other word senses is not new by itself. For instance,

[1] http://www.cs.unt.edu/ rada/senseval/. WSD systems have been evaluated in the series of Senseval workshops.

[2] http://www.cnts.ua.ac.be/conll2003/ner/

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X), pages 77–84, New York City, June 2006.

on the WSD task with respect to WordNet synsets, Kohomban and Lee (2005) trained classifiers for the top-level synsets of the WordNet semantic hierarchy, consolidating labeled examples associated with the WordNet sub-trees. To disambiguate test instances, these coarse-grained classifiers are first applied, and then fine-grained senses are determined using a heuristic mapping. By contrast, our approach does not require pre-defined relations among senses such as the WordNet hierarchy. Rather, we let the machine learning algorithm ASO automatically and implicitly find relations with respect to the disambiguation problems (i.e., finding shared predictive structure). Interestingly, in our experiments, seemingly unrelated or only loosely related word-sense pairs help to improve performance.

This paper makes two contributions. First, we present a new application of ASO to WSD. We empirically study the effective use of ASO and show that labeled examples of all the words can be effectively exploited in learning each individual disambiguator. Second, we report performance results that rival or exceed the state-of-the-art systems on Senseval lexical sample tasks.

2 Alternating structure optimization

This section gives a brief summary of ASO. We first introduce a standard linear prediction model for a single task and then extend it to a joint linear model used by ASO.

2.1 Standard linear prediction models

In the standard formulation of supervised learning, we seek a *predictor* that maps an input vector (or *feature vector*) $\mathbf{x} \in \mathcal{X}$ to the corresponding output $y \in \mathcal{Y}$. For NLP tasks, binary features are often used – for example, if the word to the left is "money", set the corresponding entry of $\mathbf{x}$ to 1; otherwise, set it to 0. A k-way classification problem can be cast as k binary classification problems, regarding output $y = +1$ and $y = -1$ as "in-class" and "out-of-class", respectively.

Predictors based on *linear prediction models* take the form: $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, where $\mathbf{w}$ is called a *weight vector*. A common method to obtain a predictor $\hat{f}$ is regularized *empirical risk minimization*, which minimizes an empirical loss of the predictor (with regularization) on the n labeled training examples $\{(\mathbf{X}_i, Y_i)\}$:

$$\hat{f} = \arg\min_f \left(\sum_{i=1}^{n} L(f(\mathbf{X}_i), Y_i) + r(f) \right) . \quad (1)$$

A *loss function* $L(\cdot)$ quantifies the difference between the prediction $f(\mathbf{X}_i)$ and the true output Y_i, and $r(\cdot)$ is a regularization term to control the model complexity.

2.2 Joint linear models for ASO

Consider m prediction problems indexed by $\ell \in \{1, \ldots, m\}$, each with n_ℓ samples $(\mathbf{X}_i^\ell, Y_i^\ell)$ for $i \in \{1, \ldots, n_\ell\}$, and assume that there exists a low-dimensional predictive structure shared by these m problems. Ando and Zhang (2005a) extend the above traditional linear model to a joint linear model so that a predictor for problem ℓ is in the form:

$$f_\ell(\Theta, \mathbf{x}) = \mathbf{w}_\ell^T\mathbf{x} + \mathbf{v}_\ell^T\Theta\mathbf{x} \,, \;\; \Theta\Theta^T = \mathbf{I} \,, \quad (2)$$

where $\mathbf{I}$ is the identity matrix. $\mathbf{w}_\ell$ and $\mathbf{v}_\ell$ are weight vectors specific to each problem ℓ. Predictive structure is parameterized by the *structure matrix* Θ shared by all the m predictors. The goal of this model can also be regarded as learning a common good feature map $\Theta\mathbf{x}$ used for all the m problems.

2.3 ASO algorithm

Analogous to (1), we compute Θ and predictors so that they minimize the empirical risk summed over all the problems:

$$[\hat{\Theta}, \{\hat{f}_\ell\}] = \arg\min_{\Theta, \{f_\ell\}} \sum_{\ell=1}^{m} \left(\sum_{i=1}^{n_\ell} \frac{L(f_\ell(\Theta, \mathbf{X}_i^\ell), Y_i^\ell)}{n_\ell} + r(f_\ell) \right) . \quad (3)$$

It has been shown in (Ando and Zhang, 2005a) that the optimization problem (3) has a simple solution using *singular value decomposition (SVD)* when we choose square regularization: $r(f_\ell) = \lambda\|\mathbf{w}_\ell\|_2^2$ where λ is a regularization parameter. Let $\mathbf{u}_\ell = \mathbf{w}_\ell + \Theta^T\mathbf{v}_\ell$. Then (3) becomes the minimization of the joint empirical risk written as:

$$\sum_{\ell=1}^{m} \left(\sum_{i=1}^{n_\ell} \frac{L(\mathbf{u}_\ell^T\mathbf{X}_i^\ell, Y_i^\ell)}{n_\ell} + \lambda\|\mathbf{u}_\ell - \Theta^T\mathbf{v}_\ell\|_2^2 \right) . \quad (4)$$

This minimization can be approximately solved by repeating the following alternating optimization procedure until a convergence criterion is met:

Nouns	art, authority, bar, bum, chair, channel, child, church, circuit, day, detention, dyke, facility, fatigue, feeling, grip, hearth, holiday, lady, material, mouth, nation, nature, post, restraint, sense, spade, stress, yew
Verbs	begin, call, carry, collaborate, develop, draw, dress, drift, drive, face, ferret, find, keep, leave, live, match, play, pull, replace, see, serve strike, train, treat, turn, use, wander wash, work
Adjectives	blind, colourless, cool, faithful, fine, fit, free, graceful, green, local, natural, oblique, simple, solemn, vital

Figure 1: Words to be disambiguated; Senseval-2 English lexical sample task.

1. Fix $(\Theta, \{\mathbf{v}_\ell\})$, and find m predictors $\{\mathbf{u}_\ell\}$ that minimizes the joint empirical risk (4).

2. Fix m predictors $\{\mathbf{u}_\ell\}$, and find $(\Theta, \{\mathbf{v}_\ell\})$ that minimizes the joint empirical risk (4).

The first step is equivalent to training m predictors independently. The second step, which couples all the predictors, can be done by setting the rows of Θ to the most significant *left singular vectors* of the *predictor (weight) matrix* $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]$, and setting $\mathbf{v}_\ell = \Theta\mathbf{u}_\ell$. That is, the structure matrix Θ is computed so that the projection of the predictor matrix $\mathbf{U}$ onto the subspace spanned by Θ's rows gives the best approximation (in the least squares sense) of $\mathbf{U}$ for the given row-dimension of Θ. Thus, intuitively, Θ captures the commonality of the m predictors.

ASO has been shown to be useful in its *semi-supervised learning* configuration, where the above algorithm is applied to a number of *auxiliary problems* that are *automatically created* from the unlabeled data. By contrast, the focus of this paper is the *multi-task learning* configuration, where the ASO algorithm is applied to a number of *real problems* with the goal of improving overall performance on these problems.

3 Effective use of ASO on word sense disambiguation

The essence of ASO is to learn information useful for prediction (predictive structure) shared by multiple tasks, assuming the existence of such shared structure. From this viewpoint, consider the target words of the Senseval-2 lexical sample task, shown in Figure 1. Here we have multiple disambiguation tasks; however, at a first glance, it is not entirely clear whether these tasks share predictive structure (or are related to each other). There is no direct semantic relationship (such as synonym or hyponym relations) among these words.

Local context	word uni-grams in 5-word window, word bi- and tri-grams of (w_{-2}, w_{-1}), $(w_{+1}, w_{+2}), (w_{-1}, w_{+1})$, $(w_{-3}, w_{-2}, w_{-1}), (w_{+1}, w_{+2}, w_{+3})$, $(w_{-2}, w_{-1}, w_{+1}), (w_{-1}, w_{+1}, w_{+2})$.
Syntactic	full parser output; see Section 3 for detail.
Global	all the words excluding stopwords.
POS	uni-, bi-, and tri-grams in 5-word window.

Figure 2: Features. w_i stands for the word at position i relative to the word to be disambiguated. The 5-word window is $[-2, +2]$. Local context and POS features are position-sensitive. Global context features are position insensitive (a bag of words).

The goal of this section is to empirically study the effective use of ASO for improving overall performance on these seemingly unrelated disambiguation problems. Below we first describe the task setting, features, and algorithms used in our implementation, and then experiment with the Senseval-2 English lexical sample data set (with the official training / test split) for the development of our methods. We will then evaluate the methods developed on the Senseval-2 data set by carrying out the Senseval-3 tasks, i.e., training on the Senseval-3 training data and then evaluating the results on the (unseen) Senseval-3 test sets in Section 4.

Task setting In this work, we focus on the Senseval *lexical sample task*. We are given a set of target words, each of which is associated with several possible senses, and their labeled instances for training. Each instance contains an occurrence of one of the target words and its surrounding words, typically a few sentences. The task is to assign a sense to each test instance.

Features We adopt the feature design used by Lee and Ng (2002), which consists of the following four types: (1) *Local context*: n-grams of nearby words (position sensitive); (2) *Global context*: all the words (excluding stopwords) in the given context (position-insensitive; a bag of words); (3) *POS*: parts-of-speech n-grams of nearby words; (4) *Syn-*

tactic relations: syntactic information obtained from parser output. To generate syntactic relation features, we use the Slot Grammar-based full parser ESG (McCord, 1990). We use as features syntactic relation types (e.g., subject-of, object-of, and noun modifier), participants of syntactic relations, and bigrams of syntactic relations / participants. Details of the other three types are shown in Figure 2.

Implementation Our implementation follows Ando and Zhang (2005a). We use a modification of the Huber's robust loss for regression: $L(p, y) = (\max(0, 1-py))^2$ if $py \geq -1$; and $-4py$ otherwise; with square regularization ($\lambda = 10^{-4}$), and perform empirical risk minimization by *stochastic gradient descent (SGD)* (see e.g., Zhang (2004)). We perform one ASO iteration.

3.1 Exploring the multi-task learning configuration

The goal is to effectively apply ASO to the set of word disambiguation problems so that overall performance is improved. We consider two factors: *feature split* and *partitioning of prediction problems*.

3.1.1 Feature split and problem partitioning

Our features described above inherently consist of four *feature groups*: local context (LC), global context (GC), syntactic relation (SR), and POS features. To exploit such a natural feature split, we explore the following extension of the joint linear model:

$$f_\ell(\{\Theta_j\}, \mathbf{x}) = \mathbf{w}_\ell^T \mathbf{x} + \sum_{j \in F} \mathbf{v}_\ell^{(j)^T} \Theta_j \mathbf{x}^{(j)} , \qquad (5)$$

where $\Theta_j \Theta_j^T = \mathbf{I}$ for $j \in F$, F is a set of disjoint feature groups, and $\mathbf{x}^{(j)}$ (or $\mathbf{v}_\ell^{(j)}$) is a portion of the feature vector $\mathbf{x}$ (or the weight vector $\mathbf{v}_\ell$) corresponding to the feature group j, respectively. This is a slight modification of the extension presented in (Ando and Zhang, 2005a). Using this model, ASO computes the structure matrix Θ_j for each feature group separately. That is, SVD is applied to the sub-matrix of the predictor (weight) matrix corresponding to each feature group j, which results in more focused dimension reduction of the predictor matrix. For example, suppose that $F = \{\mathrm{SR}\}$. Then, we compute the structure matrix Θ_{SR} from the corresponding sub-matrix of the predictor matrix $\mathbf{U}$, which is the gray region of Figure 3 (a). The structure matrices Θ_j for $j \notin F$ (associated with the white regions in the figure) should be regarded as being fixed to the zero matrices. Similarly, it is possible to compute a structure matrix from a subset of the predictors (such as noun disambiguators only), as in Figure 3 (b). In this example, we apply the extension of ASO with $F = \{\mathrm{SR}\}$ to three sets of problems (disambiguation of nouns, verbs, and adjectives, respectively) separately.

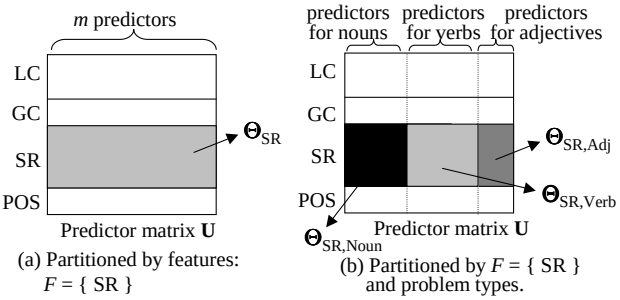


Figure 3: Examples of feature split and problem partitioning.

To see why such partitioning may be useful for our WSD problems, consider the disambiguation of "bank" and the disambiguation of "save". Since a "bank" as in "money bank" and a "save" as in "saving money" may occur in similar global contexts, certain global context features effective for recognizing the "money bank" sense may be also effective for disambiguating "save", and vice versa. However, with respect to the position-sensitive local context features, these two disambiguation problems may not have much in common since, for instance, we sometimes say "the bank announced", but we rarely say "the save announced". That is, whether problems share predictive structure may depend on feature types, and in that case, seeking predictive structure for each feature group separately may be more effective. Hence, we experiment with the configurations with and without various feature splits using the extension of ASO.

Our target words are nouns, verbs, and adjectives. As in the above example of "bank" (noun) and "save" (verb), the predictive structure of global context features may be shared by the problems irrespective of the parts of speech of the target words. However, the other types of features may be more dependent on the target word part of speech. There-

fore, we explore two types of configuration. One applies ASO to all the disambiguation problems at once. The other applies ASO separately to each of the three sets of disambiguation problems (noun disambiguation problems, verb disambiguation problems, and adjective disambiguation problems) and uses the structure matrix Θ_j obtained from the noun disambiguation problems only for disambiguating nouns, and so forth.

Thus, we explore combinations of two parameters. One is the set of feature groups F in the model (5). The other is the partitioning of disambiguation problems.

3.1.2 Empirical results

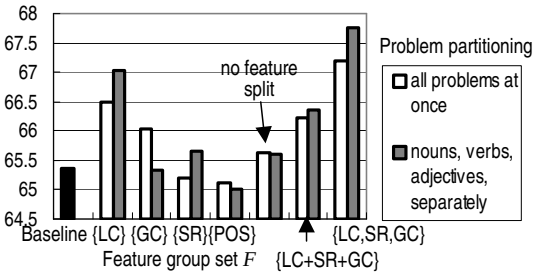


Figure 4: F-measure on Senseval-2 English test set. Multi-task configurations varying feature group set F and problem partitioning. Performance at the best dimensionality of Θ_j (in $\{10, 25, 50, 100, \cdots\}$) is shown.

In Figure 4, we compare performance on the Senseval-2 test set produced by training on the Senseval-2 training set using the various configurations discussed above. As the evaluation metric, we use the F-measure (micro-averaged)[3] returned by the official Senseval scorer. Our baseline is the standard *single-task* configuration using the same loss function (modified Huber) and the same training algorithm (SGD).

The results are in line with our expectation. To learn the shared predictive structure of local context (LC) and syntactic relations (SR), it is more advantageous to apply ASO to each of the three sets of problems (disambiguation of nouns, verbs, and adjectives, respectively), separately. By contrast, global context features (GC) can be more effectively exploited when ASO is applied to all the disambiguation problems at once. It turned out that the configuration $F = \{\mathrm{POS}\}$ does not improve the performance over the baseline. Therefore, we exclude POS from the feature group set F in the rest of our experiments. Comparison of $F = \{\mathrm{LC+SR+GC}\}$ (treating the features of these three types as one group) and $F = \{\mathrm{LC}, \mathrm{SR}, \mathrm{GC}\}$ indicates that use of this feature split indeed improves performance. Among the configurations shown in Figure 4, the best performance (67.8%) is obtained by applying ASO to the three sets of problems (corresponding to nouns, verbs, and adjectives) separately, with the feature split $F = \{\mathrm{LC}, \mathrm{SR}, \mathrm{GC}\}$.

ASO has one parameter, the dimensionality of the structure matrix Θ_j (i.e., the number of left singular vectors to compute). The performance shown in Figure 4 is the ceiling performance obtained at the best dimensionality (in $\{10, 25, 50, 100, 150, \cdots\}$). In Figure 5, we show the performance dependency on Θ_j's dimensionality when ASO is applied to all the problems at once (Figure 5 left), and when ASO is applied to the set of the noun disambiguation problems (Figure 5 right). In the left figure, the configuration $F = \{\mathrm{GC}\}$ (global context) produces better performance at a relatively low dimensionality. In the other configurations shown in these two figures, performance is relatively stable as long as the dimensionality is not too low.

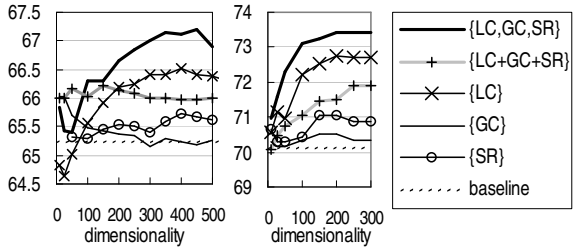


Figure 5: Left: Applying ASO to all the WSD problems at once. Right: Applying ASO to noun disambiguation problems only and testing on the noun disambiguation problems only. x-axis: dimensionality of Θ_j.

3.2 Multi-task learning procedure for WSD

Based on the above results on the Senseval-2 test set, we develop the following procedure using the feature split and problem partitioning shown in Figure 6. Let $\mathcal{N}$, $\mathcal{V}$, and $\mathcal{A}$ be sets of disambiguation problems whose target words are nouns, verbs, and adjectives, respectively. We write $\Theta_{(j,s)}$ for the struc-

[3] Our precision and recall are always the same since our systems assign exactly one sense to each instance. That is, our F-measure is the same as 'micro-averaged recall' or 'accuracy' used in some of previous studies we will compare with.

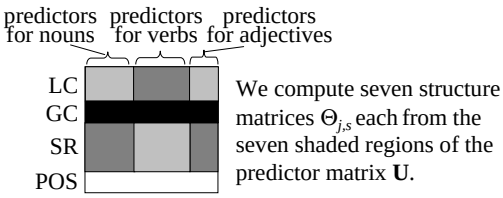


Figure 6: Effective feature split and problem partitioning.

ture matrix associated with the feature group j and computed from a problem set s. That is, we replace Θ_j in (5) with $\Theta_{(j,s)}$.

- Apply ASO to the three sets of disambiguation problems (corresponding to nouns, verbs, and adjectives), separately, using the extended model (5) with $F = \{\mathrm{LC}, \mathrm{SR}\}$. As a result, we obtain $\Theta_{(j,s)}$ for every $(j,s) \in \{\mathrm{LC}, \mathrm{SR}\} \times \{\mathcal{N}, \mathcal{V}, \mathcal{A}\}$.

- Apply ASO to all the disambiguation problems at once using the extended model (5) with $F = \{\mathrm{GC}\}$ to obtain $\Theta_{(\mathrm{GC}, \mathcal{N} \cup \mathcal{V} \cup \mathcal{A})}$.

- For a problem $\ell \in P \in \{\mathcal{N}, \mathcal{V}, \mathcal{A}\}$, our final predictor is based on the model:

$$f_\ell(\mathbf{x}) = \mathbf{w}_\ell^T \mathbf{x} + \sum_{(j,s) \in T} {\mathbf{v}_\ell^{(j,s)}}^T \Theta_{(j,s)} \mathbf{x}^{(j)} \ ,$$

  where $T = \{(\mathrm{LC}, P), (\mathrm{SR}, P), (\mathrm{GC}, \mathcal{N} \cup \mathcal{V} \cup \mathcal{A})\}$. We obtain predictor $\hat{f}_\ell$ by minimizing the regularized empirical risk with respect to $\mathbf{w}_\ell$ and $\mathbf{v}_\ell$.

We fix the dimension of the structure matrix corresponding to global context features to 50. The dimensions of the other structure matrices are set to 0.9 times the maximum possible rank to ensure relatively high dimensionality. This procedure produces 68.1% on the Senseval-2 English lexical sample test set.

3.3 Previous systems on Senseval-2 data set

Figure 7 compares our performance with those of previous best systems on the Senseval-2 English lexical sample test set. Since we used this test set for the development of our method above, our performance should be understood as the *potential performance*. (In Section 4, we will present evaluation results on the *unseen* Senseval-3 test sets.) Nevertheless, it is worth noting that our potential performance (68.1%) exceeds those of the previous best systems.

ASO multi-task learning (optimum config.)	*68.1*
classifier combination [FY02]	66.5
polynomial KPCA [WSC04]	65.8
SVM [LN02]	65.4
Our single-task baseline	*65.3*
Senseval-2 (2001) best participant	64.2

Figure 7: Performance comparison with previous best systems on Senseval-2 English lexical sample test set. FY02 (Florian and Yarowsky, 2002), WSC04 (Wu et al., 2004), LN02 (Lee and Ng, 2002)

Our single-task baseline performance is almost the same as LN02 (Lee and Ng, 2002), which uses SVM. This is consistent with the fact that we adopted LN02's feature design. FY02 (Florian and Yarowsky, 2002) combines classifiers by linear average stacking. The best system of the Senseval-2 competition was an early version of FY02. WSC04 used a polynomial kernel via the kernel Principal Component Analysis (KPCA) method (Schölkopf et al., 1998) with nearest neighbor classifiers.

4 Evaluation on Senseval-3 tasks

In this section, we evaluate the methods developed on the Senseval-2 data set above on the standard Senseval-3 lexical sample tasks.

4.1 Our methods in multi-task and semi-supervised configurations

In addition to the multi-task configuration described in Section 3.2, we test the following semi-supervised application of ASO. We first create auxiliary problems following Ando and Zhang (2005a)'s partially-supervised strategy (Figure 8) with distinct feature maps Ψ_1 and Ψ_2 each of which uses one of $\{\mathrm{LC}, \mathrm{GC}, \mathrm{SR}\}$. Then, we apply ASO to these auxiliary problems using the feature split and the problem partitioning described in Section 3.2.

Note that the difference between the multi-task and semi-supervised configurations is the source of information. The multi-task configuration utilizes the *label information* of the training examples that are labeled for the rest of the multiple tasks, and the semi-supervised learning configuration exploits a large amount of *unlabeled data*.

1. Train a classifier C_1 only using feature map Ψ_1 on the labeled data for the target task.
2. Auxiliary problems are to predict the labels assigned by C_1 to the unlabeled data, using the other feature map Ψ_2.
3. Apply ASO to the auxiliary problems to obtain Θ.
4. Using the joint linear model (2), train the final predictor by minimizing the empirical risk for fixed Θ on the labeled data for the target task.

Figure 8: Ando and Zhang (2005a)'s ASO semi-supervised learning method using partially-supervised procedure for creating relevant auxiliary problems.

	#words	#train	avg #sense per word	avg #train per sense
English	73	8611	10.7	10.0
Senseval-3 data sets				
English	57	7860	6.5	21.3
Catalan	27	4469	3.1	53.2
Italian	45	5145	6.2	18.4
Spanish	46	8430	3.3	55.5

Figure 9: Data statistics of Senseval-2 English lexical sample data set (first row) and Senseval-3 data sets. On each data set, # of test instances is about one half of that of training instances.

4.2 Data and evaluation metric

We conduct evaluations on four Senseval-3 lexical sample tasks (English, Catalan, Italian, and Spanish) using the official training / test splits. Data statistics are shown in Figure 9. On the Spanish, Catalan, and Italian data sets, we use part-of-speech information (as features) and unlabeled examples (for semi-supervised learning) provided by the organizer. Since the English data set was not provided with these additional resources, we use an in-house POS tagger trained with the PennTree Bank corpus, and extract 100K unlabeled examples from the Reuters-RCV1 corpus. On each language, the number of unlabeled examples is 5–15 times larger than that of the labeled training examples. We use syntactic relation features only for English data set. As in Section 3, we report micro-averaged F measure.

4.3 Baseline methods

In addition to the standard single-task supervised configuration as in Section 3, we test the following method as an additional baseline.

Output-based method The goal of our multi-task learning configuration is to benefit from having the labeled training examples of a number of words. An alternative to ASO for this purpose is to use directly as features the output values of classifiers trained for disambiguating the other words, which we call 'output-based method' (cf. Florian et al. (2003)). We explore several variations similarly to Section 3.1 and report the ceiling performance.

4.4 Evaluation results

Figure 10 shows F-measure results on the four Senseval-3 data sets using the official training / test splits. Both ASO multi-task learning and semi-supervised learning improve performance over the single-task baseline on all the data sets. The best performance is achieved when we combine multi-task learning and semi-supervised learning by using all the corresponding structure matrices $\Theta_{(j,s)}$ produced by both multi-task and semi-supervised learning, in the final predictors. This combined configuration outperforms the single-task supervised baseline by up to 5.7%.

Performance improvements over the supervised baseline are relatively small on English and Spanish. We conjecture that this is because the supervised performance is already close to the highest performance that automatic methods could achieve. On these two languages, our (and previous) systems outperform inter-human agreement, which is unusual but can be regarded as an indication that these tasks are difficult.

The performance of the output-based method (baseline) is relatively low. This indicates that output values or proposed labels are not expressive enough to integrate information from other predictors effectively on this task. We conjecture that for this method to be effective, the problems are required to be more closely related to each other as in Florian et al. (2003)'s named entity experiments.

A practical advantage of ASO multi-task learning over ASO semi-supervised learning is that shorter computation time is required to produce similar performance. On this English data set, training for multi-task learning and semi-supervised learning takes 15 minutes and 92 minutes, respectively, using a Pentium-4 3.20GHz computer. The computation time mostly depends on the amount of the data on which auxiliary predictors are learned. Since our experiments use unlabeled data 5–15 times larger than labeled training data, semi-supervised learning takes longer, accordingly.

	methods	English	Catalan	Italian	Spanish
ASO	multi-task learning	73.8 (+0.8)	89.5 (+1.5)	63.2 (+4.9)	89.0 (+1.0)
	semi-supervised learning	73.5 (+0.5)	88.6 (+0.6)	62.4 (+4.1)	88.9 (+0.9)
	multi-task+semi-supervised	**74.1 (+1.1)**	**89.9 (+1.9)**	**64.0 (+5.7)**	**89.5 (+1.5)**
baselines	output-based	73.0 (0.0)	88.3 (+0.3)	58.0 (-0.3)	88.2 (+0.2)
	single-task supervised learning	*73.0*	*88.0*	*58.3*	*88.0*
previous systems	SVM with LSA kernel [GGS05]	73.3	89.0	61.3	88.2
	Senseval-3 (2004) best systems	72.9 [G04]	85.2 [SGG04]	53.1 [SGG04]	84.2 [SGG04]
	inter-annotator agreement	67.3	93.1	89.0	85.3

Figure 10: Performance results on the Senseval-3 lexical sample test sets. Numbers in the parentheses are performance gains compared with the single-task supervised baseline (italicized). [G04] Grozea (2004); [SGG04] Strapparava et al. (2004).

GGS05 combined various kernels, which includes the LSA kernel that exploits unlabeled data with global context features. Our implementation of the LSA kernel with our classifier (and our other features) also produced performance similar to that of GGS05. While the LSA kernel is closely related to a special case of the semi-supervised application of ASO (see the discussion of PCA in Ando and Zhang (2005a)), our approach here is more general in that we exploit not only unlabeled data and global context features but also the labeled examples of other target words and other types of features. G04 achieved high performance on English using regularized least squares with compensation for skewed class distributions. SGG04 is an early version of GGS05. Our methods rival or exceed these state-of-the-art systems on all the data sets.

5 Conclusion

With the goal of achieving higher WSD performance by exploiting all the currently available resources, our focus was the new application of the ASO algorithm in the multi-task learning configuration, which improves performance by learning a number of WSD problems simultaneously instead of training for each individual problem independently. A key finding is that using ASO with appropriate feature / problem partitioning, labeled examples of seemingly unrelated words can be effectively exploited. Combining ASO multi-task learning with ASO semi-supervised learning results in further improvements. The fact that performance improvements were obtained consistently across several languages / sense inventories demonstrates that our approach has broad applicability and hence practical significance.

References

Rie Kubota Ando and Tong Zhang. 2005a. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853. An early version was published as IBM Research Report (2004).

Rie Kubota Ando and Tong Zhang. 2005b. High performance semi-supervised learning for text chunking. In *Proceedings of ACL-2005*.

Radu Florian and David Yarowsky. 2002. Modeling consensus: Classifier combination for word sense disambiguation. In *Proceedings of EMNLP-2002*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of CoNLL-2003*.

Cristian Grozea. 2004. Finding optimal parameter settings for high performance word sense diambiguation. In *Proceedings of Senseval-3 Workshop*.

Upali S. Kohomban and Wee Sun Lee. 2005. Learning semantic classes for word sense disambiguation. In *Proceedings of ACL-2005*.

Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of EMNLP-2002*.

Michael C. McCord. 1990. Slot Grammar: A system for simpler construction of practical natural language grammars. *Natural Language and Logic: International Scientific Symposium, Lecture Notes in Computer Science*, pages 118–145.

Bernhard Schölkopf, Alexander Smola, and Klaus-Rober Müller. 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5).

Carlo Strapparava, Alfio Gliozzo, and Claudio Giuliano. 2004. Pattern abstraction and term similarity for word sense disambiguation: IRST at Senseval-3. In *Proceedings of Senseval-3 Workshop*.

Dekai Wu, Weifeng Su, and Marine Carpuat. 2004. A kernel PCA method for superior word sense disambiguation. In *Proceedings of ACL-2004*.

Tong Zhang. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML 04*, pages 919–926.

Unsupervised Parsing with U-DOP

Rens Bod
School of Computer Science
University of St Andrews
North Haugh, St Andrews
KY16 9SX Scotland, UK
rb@dcs.st-and.ac.uk

Abstract

We propose a generalization of the supervised DOP model to unsupervised learning. This new model, which we call U-DOP, initially assigns all possible unlabeled binary trees to a set of sentences and next uses all subtrees from (a large subset of) these binary trees to compute the most probable parse trees. We show how U-DOP can be implemented by a PCFG-reduction technique and report competitive results on English (WSJ), German (NEGRA) and Chinese (CTB) data. To the best of our knowledge, this is the first paper which accurately bootstraps structure for Wall Street Journal sentences up to 40 words obtaining roughly the same accuracy as a binarized *supervised* PCFG. We show that previous approaches to unsupervised parsing have shortcomings in that they either constrain the lexical or the structural context, or both.

1 Introduction

How can we learn syntactic structure from unlabeled data in an unsupervised way? The importance of unsupervised parsing is nowadays widely acknowledged. While supervised parsers suffer from shortage of hand-annotated data, unsupervised parsers operate with unlabeled raw data, of which unlimited quantities are available. During the last few years there has been considerable progress in unsupervised parsing. To give a brief overview: van Zaanen (2000) achieved 39.2% unlabeled f-score on ATIS word strings by a sentence-aligning technique called ABL. Clark (2001) reports 42.0% unlabeled f-score on the same data using distributional clustering, and Klein and Manning (2002) obtain 51.2% unlabeled f-score on ATIS part-of-speech strings using a constituent-context model called CCM. Moreover, on Penn Wall Street Journal p-o-s-strings ≤ 10 (WSJ10), Klein and Manning (2002) report 71.1% unlabeled f-score. And the hybrid approach of Klein and Manning (2004), which combines a constituency and a dependency model, leads to a further increase of 77.6% f-score.

Although there has thus been steady progress in unsupervised parsing, all these approaches have shortcomings in that they either constrain the lexical or the structural context that is taken into account, or both. For example, the CCM model by Klein and Manning (2005) is said to describe "all contiguous subsequences of a sentence" (Klein and Manning 2005: 1410). While this is a very rich lexical model, it is still limited in that it neglects dependencies that are *non*-contiguous such as between *more* and *than* in "*BA carried more people than cargo*". Moreover, by using an "all-*substrings*" approach, CCM risks to underrepresent *structural* context. Similar shortcomings can be found in other unsupervised models.

In this paper we will try to directly model structural as well as lexical context without constraining any dependencies beforehand. An approach that may seem apt in this respect is an *all-subtrees* approach (e.g Bod 2003; Goodman 2003; Collins and Duffy 2002). Subtrees can model both contiguous and non-contiguous lexical dependencies (see section 2) and they also model constituents in a hierarchical context. Moreover, we can view the all-subtrees approach as a generalization of Klein and Manning's all-substrings approach and van Zaanen's ABL model.

In the current paper, we will use the all-subtrees approach as proposed in Data-Oriented

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 85–92, New York City, June 2006.

Parsing or DOP (Bod 1998). We will generalize the supervised version of DOP to unsupervised parsing. The key idea of our approach is to initially assign all possible unlabeled binary trees to a set of given sentences, and to next use counts of all subtrees from (a large random subset of) these binary trees to compute the most probable parse trees. To the best of our knowledge, such a model has never been tried out. We will refer to this unsupervised DOP model as *U-DOP*, while the supervised DOP model (which uses hand-annotated trees) will be referred to as *S-DOP*. Moreover, we will continue to refer to the general approach simply as *DOP*.

U-DOP is not just an engineering approach to unsupervised learning but can also be motivated from a cognitive perspective (Bod 2006): if we don't have a clue which trees should be assigned to sentences in the initial stages of language acquisition, we can just as well assume that initially all trees are possible. Only those (sub)trees that partake in computing the most probable parse trees for new sentences are actually "learned". We have argued in Bod (2006) that such an integration of unsupervised and supervised methods results in an integrated model for language learning and language use.

In the following we will first explain how U-DOP works, and how it can be approximated by a PCFG-reduction technique. Next, in section 3 we discuss a number of experiments with U-DOP and compare it to previous models on English (WSJ), German (NEGRA) and Chinese (CTB) data. To the best of our knowledge, this is the first paper which bootstraps structure for WSJ sentences up to 40 words obtaining roughly the same accuracy as a binarized *supervised* PCFG. This is remarkable since unsupervised models are clearly at a disavantage compared to supervised models which can literally reuse manually annotated data.

2 Unsupervised data-oriented parsing

At a general level, U-DOP consists of the following three steps:

1. Assign all possible binary trees to a set of sentences
2. Convert the binary trees into a PCFG-reduction of DOP
3. Compute the most probable parse tree for each sentence

Note that in unsupervised parsing we do not need to split the data into a training and a test set. In this paper, we will present results both on entire corpora and on 90-10 splits of such corpora so as to make our results comparable to a *supervised* PCFG using the treebank grammars of the same data ("*S-PCFG*").

In the following we will first describe each of the three steps given above where we initially focus on inducing trees for p-o-s strings for the WSJ10 (we will deal with other corpora and the much larger WSJ40 in section 3). As shown by Klein and Manning (2002, 2004), the extension to inducing trees for words instead of p-o-s tags is rather straightforward since there exist several unsupervised part-of-speech taggers with high accuracy, which can be combined with unsupervised parsing (see e.g. Schütze 1996; Clark 2000).

Step 1: Assign all binary trees to p-o-s strings from the WSJ10

The WSJ10 contains 7422 sentences ≤ 10 words after removing empty elements and punctuation. We assigned all possible binary trees to the corresponding part-of-speech sequences of these sentences, where each root node is labeled *S* and each internal node is labeled *X*. As an example, consider the p-o-s string NNS VBD JJ NNS, which may correspond for instance to the sentence *Investors suffered heavy losses*. This string has a total of five binary trees shown in figure 1 -- where for readability we add words as well.

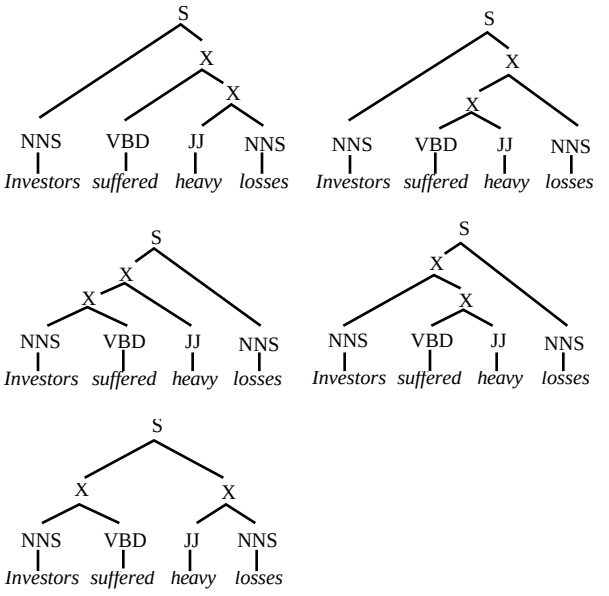


Figure 1. All binary trees for NNS VBD JJ NNS (*Investors suffered heavy losses*)

The total number of binary trees for a sentence of length n is given by the Catalan number C_{n-1}, where $C_n = (2n)!/((n+1)!\,n!)$. Thus while a sentence of 4 words has 5 binary trees, a sentence of 8 words has already 429 binary trees, and a sentence of 10 words has 4862 binary trees. Of course, we can represent the set of binary trees of a string in polynomial time and space by means of a chart, resulting in a chart-like parse forest if we also include pointers. But if we want to extract rules or subtrees from these binary trees -- as in DOP -- we need to unpack the parse forest. And since the total number of binary trees that can be assigned to the WSJ10 is almost 12 million, it is doubtful whether we can apply the unrestricted U-DOP model to such a corpus.

However, for longer sentences the binary trees are highly redundant. In these larger trees, there are many rules like $X \rightarrow XX$ which bear little information. To make parsing with U-DOP possible we therefore applied a simple heuristic which takes random samples from the binary trees for sentences ≥ 7 words before they are fed to the DOP parser. These samples were taken from the distribution of all binary trees by randomly choosing nodes and their expansions from the chart-like parse forests of the sentences (which effectively favors trees with more frequent subtrees). For sentences of 7 words we randomly sample 60% of the trees, and for sentences of 8, 9 and 10 words we sample respectively 30%, 15% and 7.5% of the trees. In this way, the set of remaining binary trees contains 8.23 * 10^5 trees, which we will refer to as the *binary tree-set*. Although it can happen that the correct tree is deleted for some sentence in the binary tree-set, there is enough redundancy in the tree-set such that either the correct binary tree can be generated by other subtrees or that a remaining tree only minimally differs from the correct tree. Of course, we may expect better results if *all* binary trees are kept, but this involves enormous computational resources which will be postponed to future research.

Step 2: Convert the trees into a PCFG-reduction of DOP

The underlying idea of U-DOP is to take all subtrees from the binary tree-set to compute the most probable tree for each sentence. Subtrees from the trees in figure 1 include for example the subtrees in figure 2 (where we again added words for readability). Note that U-DOP takes into account both contiguous and non-contiguous substrings.

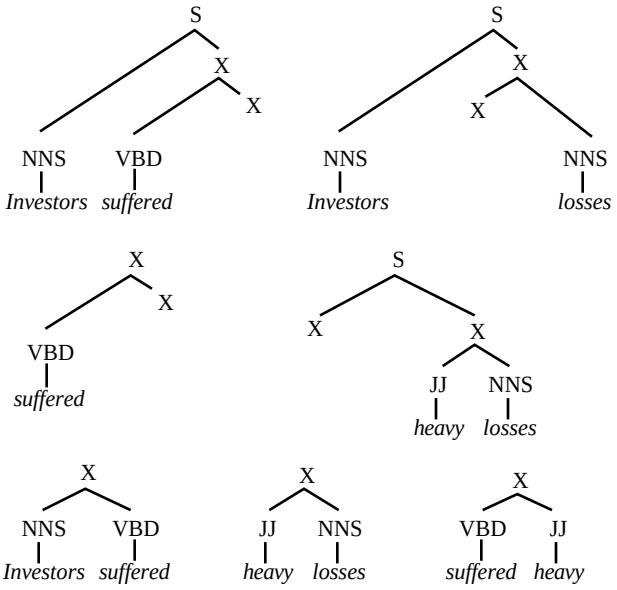


Figure 2. Some subtrees from the binary trees for NNS VBD JJ NNS given in figure 1

As in the supervised DOP approach (Bod 1998), U-DOP parses a sentence by combining corpus-subtrees from the binary tree-set by means of a leftmost node substitution operation, indicated as ∘. The probability of a parse tree is computed by summing up the probabilities of all derivations producing it, while the probability of a derivation is computed by multiplying the (smoothed) relative frequencies of its subtrees. That is, the probability of a subtree t is taken as the number of occurrences of t in the binary tree-set, $|\,t\,|$, divided by the total number of occurrences of all subtrees t' with the same root label as t. Let $r(t)$ return the root label of t:

$$P(t) = \frac{|\,t\,|}{\sum_{t':\, r(t')=r(t)} |\,t'\,|}$$

The subtree probabilities are smoothed by applying simple Good-Turing to the subtree distribution (see Bod 1998: 85-87). The probability of a derivation $t_1 \circ ... \circ t_n$ is computed by the product of the probabilities of its subtrees t_i:

$$P(t_1 \circ ... \circ t_n) = \prod_i P(t_i)$$

Since there may be distinct derivations that generate the same parse tree, the probability of a parse tree T

is the sum of the probabilities of its distinct derivations. Let t_{id} be the i-th subtree in the derivation d that produces tree T, then the probability of T is given by

$$P(T) = \Sigma_d \Pi_i P(t_{id})$$

As we will explain under step 3, the most probable parse tree of a sentence is estimated by Viterbi n-best summing up the probabilities of derivations that generate the same tree.

It may be evident that had we only the sentence *Investors suffered heavy losses* in our corpus, there would be no difference in probability between the five parse trees in figure 1, and U-DOP would not be able to distinguish between the different trees. However, if we have a different sentence where JJ NNS (*heavy losses*) appears in a different context, e.g. in *Heavy losses were reported*, its covering subtree gets a relatively higher frequency and the parse tree where *heavy losses* occurs as a constituent gets a higher total probability than alternative parse trees. Of course, it is left to the experimental evaluation whether *non*-constituents ("distituents") such as VBD JJ will be ruled out by U-DOP (section 3).

An important feature of (U-)DOP is that it considers counts of subtrees of a wide range of sizes: everything from counts of single-level rules to entire trees. A disadvantage of the approach is that an extremely large number of subtrees (and derivations) must be taken into account. Fortunately, there exists a rather compact PCFG-reduction of DOP which can also be used for U-DOP (Goodman 2003). Here we will only give a short summary of this PCFG-reduction. (Collins and Duffy 2002 show how a tree kernel can be used for an all-subtrees representation, which we will not discuss here.)

Goodman's reduction method first assigns every node in every tree a unique number which is called its address. The notation $A@k$ denotes the node at address k where A is the nonterminal labeling that node. A new nonterminal is created for each node in the training data. This nonterminal is called A_k. Let a_j represent the number of subtrees headed by the node $A@j$. Let a represent the number of subtrees headed by nodes with nonterminal A, that is $a = \Sigma_j a_j$. Goodman then gives a small PCFG with the following property: for every subtree in the training corpus headed by A, the grammar will generate an isomorphic subderivation with probability $1/a$. For a node $A@j(B@k, C@l)$, the following eight PCFG rules in figure 3 are generated, where the number in parentheses following a rule is its probability.

$A_j \rightarrow BC$	$(1/a_j)$	$A \rightarrow BC$	$(1/a)$
$A_j \rightarrow B_kC$	(b_k/a_j)	$A \rightarrow B_kC$	(b_k/a)
$A_j \rightarrow BC_l$	(c_l/a_j)	$A \rightarrow BC_l$	(c_l/a)
$A_j \rightarrow B_kC_l$	(b_kc_l/a_j)	$A \rightarrow B_kC_l$	(b_kc_l/a)

Figure 3. PCFG-reduction of DOP

In this PCFG reduction, b_k represents the number of subtrees headed by the node $B@k$, and c_l refers to the number of subtrees headed by the node $C@l$. Goodman shows by simple induction that his construction produces PCFG derivations isomorphic to (U-)DOP derivations with equal probability (Goodman 2003: 130-133). This means that summing up over derivations of a tree in DOP yields the same probability as summing over all the isomorphic derivations in the PCFG.[1]

The PCFG-reduction for U-DOP is slightly simpler than in figure 3 since the only labels are S and X, and the part-of-speech tags. For the tree-set of $8.23 * 10^5$ binary trees generated under step 1, Goodman's reduction method results in a total number of $14.8 * 10^6$ distinct PCFG rules. While it is still feasible to parse with a rule-set of this size, it is evident that our approach can deal with longer sentences only if we further reduce the size of our binary tree-set.

It should be kept in mind that while the probabilities of all parse trees generated by DOP sum up to 1, these probabilities do not converge to the "true" probabilities if the corpus grows to infinity (Johnson 2002). In fact, in Bod et al. (2003) we showed that the most probable parse tree as defined above has a tendency to be constructed by the *shortest derivation* (consisting of the fewest and thus largest subtrees). A large subtree is overruled only if the combined relative frequencies of smaller subtrees yields a larger score. We refer to Zollmann and Sima'an (2005) for a recently proposed estimator that *is* statistically consistent (though it is not yet known how this estimator performs on the WSJ) and to Zuidema (2006) for a theoretical comparison of existing estimators for DOP.

[1] As in Bod (2003) and Goodman (2003: 136), we additionally use a correction factor to redress DOP's bias discussed in Johnson (2002).

Step 3: Compute the most probable parse tree for each WSJ10 string

While Goodman's reduction method allows for efficiently computing the most probable derivation for each sentence (i.e. the Viterbi parse), it does not allow for an efficient computation of (U-)DOP's most probable parse tree since there may be exponentially many derivations for each tree whose probabilities have to be summed up. In fact, the problem of computing the most probable tree in DOP is known to be NP hard (Sima'an 1996). Yet, the PCFG reduction in figure 4 can be used to *estimate* DOP's most probable parse tree by a Viterbi *n*-best search in combination with a CKY parser which computes the *n* most likely derivations and next sums up the probabilities of the derivations producing the same tree. (We can considerably improve efficiency by using *k*-best hypergraph parsing as recently proposed by Huang and Chiang 2005, but this will be left to future research).

In this paper, we estimate the most probable parse tree from the 100 most probable derivations (at least for the relatively small WSJ10). Although such a heuristic does not guarantee that the most probable parse is actually found, it is shown in Bod (2000) to perform at least as well as the estimation of the most probable parse with Monte Carlo techniques. However, in computing the 100 most probable derivations by means of Viterbi it is prohibitive to keep track of all subderivations at each edge in the chart. We therefore use a pruning technique which deletes any item with a probability less than 10^{-5} times of that of the best item from the chart.

To make our parse results comparable to those of Klein and Manning (2002, 2004, 2005), we will use exactly the same evaluation metrics for unlabeled precision (UP) and unlabeled recall (UR), defined in Klein (2005: 21-22). Klein's definitions slightly differ from the standard PARSEVAL metrics: multiplicity of brackets is ignored, brackets of span one are ignored and the bracket labels are ignored. The two metrics of UP and UR are combined by the unlabled f-score F1 which is defined as the harmonic mean of UP and UR: F1 = 2*UP*UR/(UP+UR). It should be kept in mind that these evaluation metrics were clearly inspired by the evaluation of *supervised* parsing which aims at mimicking *given* tree annotations as closely as possible. Unsupervised parsing is different in this respect and it is questionable whether an evaluation on a pre-annotated corpus such as the WSJ is the most appropriate one. For a subtle discussion on this issue, see Clark (2001) or Klein (2005).

3 Experiments

3.1 Comparing U-DOP to previous work

Using the method described above, our parsing experiment with all p-o-s strings from the WSJ10 results in an f-score of 78.5%. We next tested U-DOP on two additional domains from Chinese and German which were also used in Klein and Manning (2002, 2004): the Chinese treebank (Xue et al. 2002) and the NEGRA corpus (Skut et al. 1997). The CTB10 is the subset of p-o-s strings from the Penn Chinese treebank containing 10 words or less after removal of punctuation (2437 strings). The NEGRA10 is the subset of p-o-s strings of the same length from the NEGRA corpus using the supplied converson into Penn treebank format (2175 strings). Table 1 shows the results of U-DOP in terms of UP, UR and F1 compared to the results of the CCM model by Klein and Manning (2002), the DMV dependency learning model by Klein and Manning (2004) together with their combined model DMV+CCM.

Model	English (WSJ10)			German (NEGRA10)			Chinese (CTB10)		
	UP	UR	F1	UP	UR	F1	UP	UR	F1
CCM	64.2	81.6	71.9	48.1	85.5	61.6	34.6	64.3	45.0
DMV	46.6	59.2	52.1	38.4	69.5	49.5	35.9	**66.7**	**46.7**
DMV+CCM	69.3	88.0	77.6	49.6	89.7	63.9	33.3	62.0	43.3
U-DOP	**70.8**	**88.2**	**78.5**	**51.2**	**90.5**	**65.4**	**36.3**	64.9	46.6

Table 1. Results of U-DOP compared to previous models on the same data

Table 1 indicates that our model scores slightly better than Klein and Manning's combined DMV+CCM model, although the differences are small (note that for Chinese the single DMV model scores better than the combined model and slightly better than U-DOP). But where Klein and Manning's combined model is based on both a constituency and a dependency model, U-DOP is, like CCM, only based on a notion of constituency. Compared to CCM alone, the all-subtrees approach employed by U-DOP shows a clear improvement (except perhaps for Chinese). It thus seems to pay off to use all subtrees rather than just all (contiguous) substrings in bootstrapping

constituency. It would be interesting to investigate an extension of U-DOP towards dependency parsing, which we will leave for future research. It is also noteworthy that U-DOP does not employ a separate class for non-constituents, so-called distituents, while CCM does. Thus good results can be obtained without keeping track of distituents but by simply assigning all binary trees to the strings and letting the DOP model decide which substrings are most likely to form constituents.

To give an idea of the constituents learned by U-DOP for the WSJ10, table 2 shows the 10 most frequently constituents in the trees induced by U-DOP together with the 10 actually most frequently occurring constituents in the WSJ10 and the 10 most frequently occurring part-of-speech sequences (bigrams) in the WSJ10.

Rank	Most frequent U-DOP constituents	Most Frequent WSJ10 constituents	Most frequent WSJ10 substrings
1	DT NN	DT NN	NNP NNP
2	NNP NNP	NNP NNP	DT NN
3	DT JJ NN	CD CD	JJ NN
4	IN DT NN	JJ NNS	IN DT
5	CD CD	DT JJ NN	NN IN
6	DT NNS	DT NNS	DT JJ
7	JJ NNS	JJ NN	JJ NNS
8	JJ NN	CD NN	NN NN
9	VBN IN	IN NN	CD CD
10	VBD NNS	IN DT NN	NN VBZ

Table 2. Most frequently learned constituents by U-DOP together with most frequently occurring constituents and p-o-s sequences (for WSJ10)

Note that there are no distituents among U-DOP's 10 most frequently learned constituents, whilst the third column shows that distituents such as IN DT or DT JJ occur very frequently as substrings in the WSJ10. This may be explained by the fact that (the constituent) DT NN occurs more frequently as a substring in the WSJ10 than (the distituent) IN DT, and therefore U-DOP's probability model will favor a covering subtree for IN DT NN which consists of a division into IN *X* and DT NN rather than into IN DT and *X* NN, other things being equal. The same kind reasoning can be made for a subtree for DT JJ NN where the constituent JJ NN occurs more frequently as a substring than the distituent DT JJ. Of course the situation is somewhat more complex in DOP's sum-of-products model, but our argument may illustrate why distituents like IN DT or DT JJ are not proposed among the most frequent constituents by U-DOP while larger constituents like IN DT NN and DT JJ NN are in fact proposed.

3.2 Testing U-DOP on held-out sets and longer sentences (up to 40 words)

We were also interested in U-DOP's performance on a held-out test set such that we could compare the model with a *supervised* PCFG treebank grammar trained and tested on the same data (S-PCFG). We started by testing U-DOP on 10 different 90%/10% splits of the WSJ10, where 90% was used for inducing the trees, and 10% to parse new sentences by subtrees from the binary trees from the training set (or actually a PCFG-reduction thereof). The supervised PCFG was right-binarized as in Klein and Manning (2005). The following table shows the results.

Model	UP	UR	F1
U-DOP	70.6	**88.1**	78.3
S-PCFG	**84.0**	79.8	**81.8**

Table 3. Average f-scores of U-DOP compared to a supervised PCFG (S-PCFG) on 10 different 90-10 splits of the WSJ10

Comparing table 1 with table 3, we see that on 10 held-out WSJ10 test sets U-DOP performs with an average f-score of 78.3% (SD=2.1%) only slightly worse than when using the entire WSJ10 corpus (78.5%). Next, note that U-DOP's results come near to the average performance of a binarized supervised PCFG which achieves 81.8% unlabeled f-score (SD=1.8%). U-DOP's unlabeled recall is even higher than that of the supervised PCFG. Moreover, according to paired *t*-testing, the differences in f-scores were *not* statistically significant. (If the PCFG was not post-binarized, its average f-score was 89.0%.)

As a final test case for this paper, we were interested in evaluating U-DOP on WSJ sentences ≤ 40 words, i.e. the WSJ40, which is with almost 50,000 sentences a much more challenging test case than the relatively small WSJ10. The main problem for U-DOP is the astronomically large number of possible binary trees for longer sentences, which therefore need to be even more heavily pruned than before.

We used a similar sampling heuristic as in section 2. We started by taking 100% of the trees for sentences ≤ 7 words. Next, for longer sentences we reduced this percentage with the relative increase of the Catalan number. This effectively means that we randomly selected the same number of trees for each sentence ≥ 8 words, which is 132 (i.e. the

number of possible binary trees for a 7-word sentence). As mentioned in section 2, our sampling approach favors more frequent trees, and trees with more frequent subtrees. The binary tree-set obtained in this way for the WSJ40 consists of $5.11 * 10^6$ different trees. This resulted in a total of 88+ million distinct PCFG rules according to the reduction technique in section 2. As this is the largest PCFG we have ever attempted to parse with, it was prohibitive to estimate the most probable parse tree from 100 most probable derivations using Viterbi *n*-best. Instead, we used a beam of only 15 most probable derivations, and selected the most probable parse from these. (The number 15 is admittedly ad hoc, and was inspired by the performance of the so-called SL-DOP model in Bod 2002, 2003). The following table shows the results of U-DOP on the WSJ40 using 10 different 90-10 splits, compared to a supervised binarized PCFG (S-PCFG) and a supervised binarized DOP model (S-DOP) on the same data.

Model	F1
U-DOP	64.2
S-PCFG	64.7
S-DOP	81.9

Table 4. Performance of U-DOP on WSJ40 using10 different 90-10 splits, compared to a binarized S-PCFG and a binarized S-DOP model.

Table 4 shows that U-DOP obtains about the same results as a binarized supervised PCFG on WSJ sentences ≤ 40 words. Moreover, the differences between U-DOP and S-PCFG were not statistically significant. This result is important as it shows that it is possible to parse the rather challinging WSJ in a completely *unsupervised* way obtaining roughly the same accuracy as a *supervised* PCFG. This seems to be in contrast with the CCM model which quickly degrades if sentence length is increased (see Klein 2005). As Klein (2005: 97) notes, CCM's strength is finding common short constituent chunks. U-DOP on the other hand has a preference for large (even largest possible) constituent chunks. Klein (2005: 97) reports that the combination of CCM and DMV seems to be more stable with increasing sentence length. It would be extremely interesting to see how DMV+CCM performs on the WSJ40.

It should be kept in mind that simple treebank PCFGs do not constitute state-of-the-art supervised parsers. Table 4 indicates that U-DOP's performance remains still far behind that of S-DOP (and indeed of other state-of-the-art supervised parsers such as Bod 2003 or Charniak and Johnson 2005). Moreover, if S-DOP is not post-binarized, its average f-score on the WSJ40 is 90.1% -- and there are some hybrid DOP models that obtain even higher scores (see Bod 2003). Our long-term goal is to try to outperform S-DOP by U-DOP. An important advantage of U-DOP is of course that it only needs unannotated data of which unlimited quanitities are available. Thus it would be interesting to test how U-DOP performs if trained on e.g. 100 times more data. Yet, as long as we compute our f-scores on *hand*-annotated data like Penn's WSJ, the S-DOP model is clearly at an advantage. We therefore plan to compare U-DOP and S-DOP (and other supervised parsers) in a concrete application such as phrase-based machine translation or as a language model for speech recognition.

4 Conclusions

We have shown that the general DOP approach can be generalized to unsupervised learning, effectively leading to a single model for both supervised and unsupervised parsing. Our new model, U-DOP, uses all subtrees from (in principle) all binary trees of a set of sentences to compute the most probable parse trees for (new) sentences. Although heavy pruning of trees is necessary to make our approach feasible in practice, we obtained competitive results on English, German and Chinese data. Our parsing results are similar to the performance of a binarized supervised PCFG on the WSJ ≤ 40 sentences. This triggers the provocative question as to whether it is possible to beat supervised parsing by unsupervised parsing. To cope with the problem of evaluation, we propose to test U-DOP in specific applications rather than on hand-annotated data.

References

Bod, R. 1998. *Beyond Grammar: An Experience-Based Theory of Language*, Stanford: CSLI Publications (Lecture notes number 88), distributed by Cambridge University Press.

Bod, R. 2000. An improved parser for data-oriented lexical-functional analysis. *Proceedings ACL'2000*, Hong Kong.

Bod, R. 2002. A unified model of structural organization in language and music. *Journal of*

Artificial Intelligence Research 17(2002), 289-308.

Bod, R., R. Scha and K. Sima'an (eds.) 2003. *Data-Oriented Parsing*. CSLI Publications/University of Chicago Press.

Bod, R. 2003. An efficient implementation of a new DOP model. *Proceedings EACL'2003*, Budapest.

Bod, R. 2006. Exemplar-based syntax: How to get productivity from examples? *The Linguistic Review* 23(3), Special Isssue on Exemplar-Based Models in Linguistics.

Charniak, E. and M. Johnson 2005. Coarse-to-fine n-best parsing and Max-Ent discriminative reranking. *Proceedings ACL'2005*, Ann-Arbor.

Clark, A. 2000. Inducing syntactic categories by context distribution clustering. *Proceedings CONLL'2000*.

Clark, A. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. *Proceedings CONLL'2001*.

Collins, M. and N. Duffy 2002. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. *Proceedings ACL'2002*, Philadelphia.

Goodman, J. 2003. Efficient algorithms for the DOP model. In R. Bod, R. Scha and K. Sima'an (eds.). *Data-Oriented Parsing*, The University of Chicago Press.

Huang, L. and Chiang D. 2005. Better *k*-best parsing. *Proceedings IWPT'2005*, Vancouver.

Johnson, M. 2002. The DOP estimation method is biased and inconsistent. *Computational Linguistics* 28, 71-76.

Klein, D. 2005. *The Unsupervised Learning of Natural Language Structure*. PhD thesis, Stanford University.

Klein, D. and C. Manning 2002. A general constituent-context model for improved grammar induction. *Proceedings ACL'2002*, Philadelphia.

Klein, D. and C. Manning 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. *Proceedings ACL'2004*, Barcelona.

Klein, D. and C. Manning 2005. Natural language grammar induction with a generative constituent-context model. *Pattern Recognition* 38, 1407-1419.

Schütze, H. 1995. Distributional part-of-speech tagging. *Proceedings ACL'1995*, Dublin.

Sima'an, K. 1996. Computational complexity of probabilistic disambiguation by means of tree grammars. *Proceedings COLING'1996*, Copenhagen.

Skut, W., B. Krenn, T. Brants and H. Uszkoreit 1997. An annotation scheme for free word order languages. *Proceedings ANLP'97*.

Xue, N., F. Chiou and M. Palmer 2002. Building a large-scale annotated Chinese corpus. *Proceedings COLING 2002*, Taipei.

van Zaanen, M. 2000. ABL: Alignment-Based Learning. *Proceedings COLING'2000*, Saarbrücken.

Zollmann, A. and K. Sima'an 2005. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*, in press.

Zuidema, W. 2006. Theoretical evaluation of estimation methods for data-oriented parsing. *Proceedings EACL'2006*, Trento.

A Lattice-Based Framework for Enhancing Statistical Parsers with Information from Unlabeled Corpora

Michaela Atterer
Institute for NLP
University of Stuttgart, Germany
atterer@ims.uni-stuttgart.de

Hinrich Schütze
Institute for NLP
University of Stuttgart, Germany
hinrich@hotmail.com

Abstract

Great strides have been made in building statistical parsers trained on annotated corpora such as the Penn treebank. However, recently performance improvements have leveled off. New information sources need to be considered to make further progress in parsing. In this paper, we propose a new method of using unlabeled corpora for improving syntactic disambiguation. The method is tested on the problem of relative clause attachment with encouraging results.

1 Introduction

Great strides have been made in building statistical parsers trained on annotated corpora such as the Penn treebank (Marcus et al., 1993). However, recently performance improvements have leveled off (Bikel, 2004; Collins and Koo, 2005; Klein and Manning, 2003; Charniak and Johnson, 2005). New information sources need to be considered to make further progress in parsing. One information source that is available in virtually unlimited quantity is unlabeled text. As a large body of work on unsupervised learning from corpora has shown, there is valuable syntactic and semantic information in natural language even if it is unlabeled. We propose to combined supervised and unsupervised learning for syntactic disambiguation as sketched in Figure 1. In the supervised phase, a probabilistic parser is trained on a labeled corpus. In the unsupervised phase, the parser is enriched with information from a large unlabeled corpus.

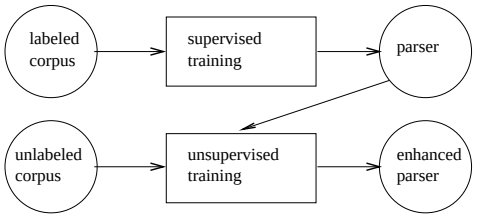


Figure 1: The proposed framework for combining supervised and unsupervised learning.

Exploiting unlabeled resources is of particular importance when training sets are small. Training sets are expensive and thus a major obstacle for broad deployment of statistical NLP methods. Statistical methods have to be adapted to new languages and new domains (e.g., a parser trained on WSJ will not work well on manuals). In many practical settings, training sets available during adapation will be small due to the high cost of training set creation. This motivates us to study the effect of *training set size* on the performance of the method proposed here. Since training sets cannot be assumed to be large in general, it is important to investigate whether methods are still applicable when training sets are smaller than the standard sets used in the research community.

There is a long tradition of using structural analysis of unlabeled corpora for syntactic disambiguation (e.g., (Hindle and Rooth, 1991)). One of the contributions of this paper is a general framework for using unsupervised acquisition of lexical information for structural dis-

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 93–100, New York City, June 2006.

ambiguation. This framework is based on lexical dependencies because they are mostly local and can therefore be extracted reliably from unlabeled text. At the same time, these extracted dependencies can be easily incorporated into the trained parser. Dependencies are thus well-suited to serve as the common currency for integrating information in combined supervised and unsupervised learning.

2 Structural Disambiguation

Conceptually, we would like to factor the parsing problem into decisions that can be made on purely structural grounds (e.g., recognition of base NPs) and more difficult attachment decisions, in particular those that require world knowledge, e.g. in Example 1

(1) Mr. Baker found [an opening] under [the house] that *led to* a fume-filled coal mine.

Does the opening lead to the coal mine, or does the house? We make the simplifying assumption that "semantic" attachment decisions are independent of each other. This is often the case on a purely syntactic level although it is clearly not true semantically since semantically inconsistent attachments can give rise to incoherent readings.

We formalize an attachment ambiguity as a phrase XP having two or more possible attachment points $i_1, i_2, \ldots$ in a sentence S. Let R be the parse of a sentence S with XP removed. To make an attachment decision, we form triples of the form $< \mathrm{R}, i, \mathrm{XP} >$ where i is a possible attachment node for XP in R. We define a set of generalization functions $G = \{g_j\}$ that map triples into more general triples. Some functions simply delete material, e.g., the subject of the sentence. Others replace nouns with their classes, e.g., "Canada" with "country". Each g_j modifies either R or XP, but not both. The functions can be applied in any order. Functions g_j that would delete the node i are not permitted.

We define a subsumption relationship $\subseteq$ on the set of triples produced from $< \mathrm{R}, i, \mathrm{XP} >$: $< T_1, i, Y_1 > \subseteq < T_2, i, Y_2 >$ iff $T_1 \subseteq T_2$ and $Y_1 \subseteq Y_2$, where a phrase structure tree P_1 is subsumed by P_2 iff the nodes of P_1 can be mapped onto P_2 preserving dominance and if nodes are mapped onto identical nodes or more specific nodes (e.g., "country" onto "Canada"). All g_j obey the constraint $g_j(< \mathrm{R}, i, \mathrm{XP} >) \subseteq < \mathrm{R}, i, \mathrm{XP} >$.

Triples are evaluated by an evaluation function ϕ that assesses the support of the lexical relationships in the triple in the unlabeled corpus C: $\phi(< \mathrm{R}, i, \mathrm{XP} >) \in \mathcal{R}$. Generalization is necessary because the particular set of words found in a sentence will rarely occur in C – and even if it does we don't know what the correct parse of the sentence is. The functions g_j produce a series of more and more abstract triples so as to guarantee that C contains enough data for evaluation.

The measure we use here to evaluate triples is pointwise mutual information with respect to an unlabeled corpus C. We define:

$$\begin{aligned} \phi(< \mathrm{T}, i, \mathrm{Y} >) &= MI(< \mathrm{T}, i, \mathrm{Y} >) \\ &= \log_2 \frac{P(< \mathrm{T}, i, \mathrm{Y} >)}{P(\mathrm{T})P(\mathrm{Y})} \end{aligned}$$

$$\text{for } P(< \mathrm{T}, i, \mathrm{Y} >), P(\mathrm{T}), P(\mathrm{Y}) \neq 0$$

$$\phi(< \mathrm{T}, i, \mathrm{Y} >) = 0 \qquad \text{otherwise}$$

where the probabilities are estimated on the unlabeled corpus C. $P(\mathrm{T})$ and $P(\mathrm{Y})$ are the probabilities of dependency structures T and Y occurring in C and $P(< \mathrm{T}, i, \mathrm{Y} >)$ is the probability of the dependency structures of T and Y, with Y attached at node i in T, occurring in C.

The set of triples $Q(< \mathrm{R}, i, \mathrm{XP} >, G)$ derived from $< \mathrm{R}, i, \mathrm{XP} >$ by successive applications of one, two or more generalization functions $g_j \in G$ forms a lattice with respect to $\subseteq$. $< \mathrm{R}, i, \mathrm{XP} >$ is the supremum and $< \emptyset, i, \emptyset >$ the infimum of this lattice. An example of such a lattice is shown in Figure 2 (see below for more detailed discussion). $\phi(< \emptyset, i, \emptyset >)$ is defined as a constant, which depends on the disambiguation task at hand. We take advantage of the lattice structure to compute the *affinity* A between R and XP which expresses to what extent attachment of XP in R at node i is supported by lexical dependencies in C. We propose three different definitions of A:

- The maximum with respect to $<$ on $\mathcal{R}$: $A_< = \max_<(\{\phi(q) | q \in Q\})$

- The sum over the lattice: $A_\Sigma = \sum_{q \in Q} \phi(q)$
- The MI of the maximum with respect to $\subseteq$: $A_\subseteq = \phi(\max_\subseteq(\{q | q \in Q, \phi(q) \neq 0\}))$ (if there are several maximal q, we take the average of their MI values)

Intuitively, we are searching for evidence in C that XP and R fit well together like a key and a lock. Affinity measure $A_<$ selects the best fitting generalization of the triple whereas A_Σ considers the joint evidence of all triples. Maximum and sum can only be computed if the lattice is small. Measure $A_\subseteq$ has the advantage of circumventing the need of computing the entire lattice. We move down from the original triple until we find a "layer" of the lattice where probabilities are not zero. In this paper, we only report results for $A_<$.

The actual syntactic disambiguation is performed by comparing the affinities $A(Q(< \mathrm{R}, i_k, \mathrm{XP} >))$ for the possible attachment nodes $i_1, i_2, \ldots$ and selecting the node with the highest affinity.

3 Experimental Setup

When computing the mutual information of an attachment constellation, the required probabilities are estimated based on dependency parses of the unlabeled corpus produced by Minipar (Lin, 1998), a dependency parser that recognizes a wide range of dependencies. We use the Reuters RCV1 corpus (Lewis et al., 2004) as our unlabeled corpus. The first 50 weeks (about 80,000,000 words) were parsed with Minipar and dependencies stored in an inverted index for easy querying. The inverted index is implemented using Lucene (Lucene, 2006). This setup enables searching for the frequency of lexical dependencies. For example, we can query for the number of times that *cat* was the subject of *chase*, and we can estimate the probabilities $P(T)$, $P(Y)$, and $P(< T, i, Y >)$ as relative frequencies by counting the number of times the corresponding dependency structures occur in the corpus. A constellation (T, Y, or $< T, i, Y >$) is first represented as a dependency structure and, for reasons of efficiency, the number of occurrences of this dependency structure is then approximated as the number of sentences that contain all binary dependencies in the structure. We take a trained parser (Minipar[1] or the Collins parser, depending on the experiment), run it on Penn Treebank sentences, search for the type of attachment ambiguity we are interested in and, if it occurs, present two triples of the form $< \mathrm{R}, i, \mathrm{XP} >$ and $< \mathrm{R}, j, \mathrm{XP} >$ to the disambiguation component, where i and j are two possible attachment sites for XP in R. Sections 00–12 of the WSJ were used as the development set, and sections 13–24 as the test set.

4 Application to Relative Clause Attachment

Sentence 1 is a typical example of relative clause (RC) attachment ambiguity.

Both attachments are grammatical, but intuitively *opening* is more likely to occur with the verbs *lead* or *lead to* than *house*. Our hypothesis is that this type of pragmatic knowledge (openings lead to something, houses don't) will be reflected in dependencies extracted from a large corpus. Extracting dependencies is particularly important as RC attachment is a more difficult problem than PP attachment as the following examples show.

(2) Texaco Inc. reported [an 11% increase] in [third-quarter earnings], which it *attributed* partly to the company's massive restructuring [...]

(3) Earlier this year DPC Acquisition made [a $15-a-share offer] for [Dataproducts], which the Dataproducts board said it *rejected* [...]

(4) [...] said Edmar Mednis, [the expert commentator] for [the match], which was *attended* by hundreds of chess fans.

RC attachment interacts with a wider range of grammatical phenomena than PP attachment (e.g., object vs. subject relatives, passive, and agreement). Also, many cases of PP attachment can be resolved structurally. For example, an

[1] Minipar attaches relative clauses low by default, resulting in many incorrect attachment decisions. Since relative clauses are rare, we do not systematically eliminate them when computing "unlabeled" statistics.

on-PP after *rely* almost always attaches to the verb. In contrast, RC attachment is mostly semantic (e.g., *opening* is a more typical subject of *lead to* than *house*). For our experiments, we extracted all sentences from the WSJ corpus that contained a pattern of the form `NP1 Prep NP2 which/that`. (See (Web Appendix, 2006) for documentation on the patterns used.) Our development set contained 282 *which*-clauses (71 with high attachment; 211 with low attachment) and 385 *that*-clauses (156 with high attachment and 229 with low attachment). The test corpus contained 264 *which*-clauses (71 with high attachment and 193 with low attachment) and 391 *that*-clauses (175 with high attachment and 216 with low attachment). For the case of relative clause attachment, we simplify the representation of triples $< \mathrm{R}, i_1, \mathrm{XP} >, < \mathrm{R}, i_2, \mathrm{XP} >$ to pairs $< NP_1, \mathrm{XP} >$, $< NP_2, \mathrm{XP} >$,where NP_1 and NP_2 are two potential attachment sites the relative clause can attach to, and XP consists of verb and object (if there is an object) of the relative clause. The maximum lattice for relative clause attachment is depicted in Figure 2. The lattice will be smaller if there is no object, premodifying adjective etc. The supremum of the lattice corresponds to a query that includes the entire NP (including modifying adjectives/nouns)[2], the verb and its object: *"weekly mod report" && "report subj show" && "decline obj show"*. The generalizing options are:

- strip the NP of the modifying adjective/noun (*weekly report* → *report*)
- use only the head noun of the NP (*Catastrophic Care Act* → *Act*)
- use the head noun in lower case (*Act* → *act*)
- for named entities use a hypernym of the NP (*American Bell Telephone Co.* → *company*)
- strip the object from XP (*company have subsidiary* → *company have*)
- don't use any context at all. In this case the default attachment (to the last NP) is selected.

To compute the values of ϕ, we first parse the sentence with Minipar and extract the relevant verb and grammatical relation. Then we query the database for *subject*, *object*, and *modifier* relations to calculate $P(\mathrm{NP})$, $P(\mathrm{XP})$, and $P(< \mathrm{NP}, \mathrm{XP} >)$. For example, $P(<$ *opening*, *lead_to* $>)$ is estimated based on the query *"opening subj lead_to"*. Including further information about the context (e.g. about the object of the verb in the relative clause) – as opposed to only using noun-verb co-occurrence – proved particularly useful for light verbs like *make* and *have*.

[2]From the Minipar output, we use all adjectives which modify the NP via the relation *mod*, and all nouns, which modify the NP via the relation *nn*.

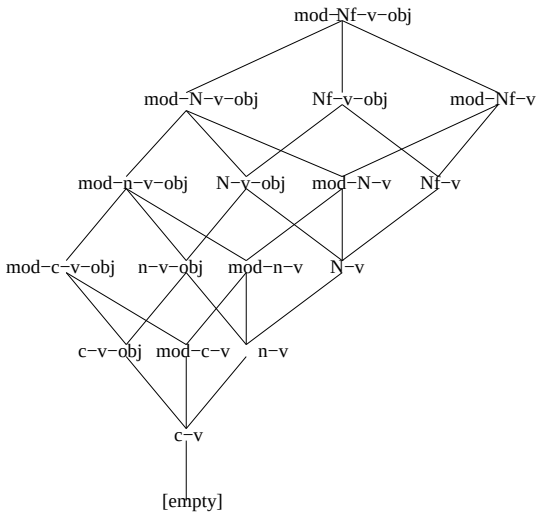


Figure 2: Partially ordered set of pairs of potential attachment site NP and relative clause XP, where mod: premodifying adjective or noun, Nf: head noun with lexical modifiers, N: head noun only, n: head noun in lower case, c: class of NP, v: verb in relative clause, obj: object of verb in the relative clause.

4.1 Named Entities

Named entities often cause sparse data problems. For this reason, we also use queries in lower case and queries where the named entity is replaced by its class. For Example 5 we would have queries *Act subj boost* and *act subj boost*.

(5) Congress still is struggling to dismantle [the unpopular Catastrophic Care Act] of [1988], which boosted benefits for the elderly and taxed them to pay for the new coverage.

To identify the class of a named entity we use LingPipe (LingPipe, 2006). When LingPipe identifies a named entity as a company or organization, we replace it with *company* in the query. Locations are replaced by *country*. Persons block RC attachment because neither *which* nor *that* clauses attach to person names, resulting in an attachment of the RC to the other NP.

4.2 A Worked Example

Table 1 shows mutual information values for the queries constructed for sentence 6.

(6) The firmness in heating oil was attributed to colder weather in parts of the U.S. and to the latest [weekly report] by [the American Petroleum Institute], which *showed* a decline in inventories of the fuel.

queries for <weekly report, show decline> etc.	MI
"weekly mod report" && "report subj show" && "decline obj show"	0
"weekly mod report" && "report subj show"	8.63
"report subj show" && "decline obj show"	5.38
"report subj show"	7.21

queries for <API, show decline> etc.	MI
"American_Petroleum_Institute subj show" && "decline obj show"	8.44
"Institute subj show" && "decline obj show"	0
"institute subj show" && "decline obj show"	0
"company subj show" && "decline obj show"	1.39
"American_Petroleum_Institute subj show"	8.47
"Institute subj show"	0
"institute subj show"	4.50
"company subj show"	3.17
[empty]	6

Table 1: Queries for computing $P(< NP_1, \mathrm{XP} >)$ (high attachment, above) and $P(< NP_2, \mathrm{XP} >)$ (low attachment, below) for Example 6, (including further tuples after applying g_j) and corresponding mutual information values (MI).

In Table 1, the highest value for the high attachment site *weekly report* is 8.63 and the highest value for the low attachment site is 8.47. We hence choose high attachment for this case. Note that the low attachment site has a value 6 for the empty context. This value reflects the bias for low attachment: the majority of relative clauses are attached low. If all MI-values are zero or otherwise low, this procedure will automatically result in low attachment.

4.3 Decision list

For increased accuracy, the structural disambiguation method is embedded in the following decision list. Step 4 is the lattice-based algorithm described above.

1. If Minipar has already chosen high attachment, choose high attachment (only relevant for named entities in some of the *which* clauses in our data).
2. If there is agreement between the verb and only one of the NPs, attach to this NP.
3. If one of the NPs is in a list of person entities, attach to the other NP.[3]
4. If possible, use structural disambiguation based on the affinities computed on the Reuters corpus.
5. If none of the above strategies was successful (e.g. in the case of parsing errors, where the verb or the relation cannot be retrieved), attach low.

5 Evaluation

that clauses	accuracy
development set, baseline	59.48%
development set, algorithm	64.42%
test set, baseline	55.24%
test set, algorithm	60.87%
which clauses	accuracy
development set, baseline	74.82%
development set, Minipar	78.37%
development set, algorithm	82.27%
test set, baseline	73.12%
test set, Minipar	75.75%
test set, algorithm	78.41%

Table 2: Evaluation results (percentage of correct attachments) for *that* and *which* clauses.

We first evaluated the accuracy of relative clause attachment with Minipar as the base parser. Table 2 shows the evaluation results

[3]This list contains 136 entries and was semiautomatically computed from the Reuters corpus: Antecedents of *who* relative clauses were extracted, and the top 200 were filtered manually.

when the algorithm is run against our development and test sets. We set $\phi(< \emptyset, i, \emptyset >) = 6$.[4] The baseline is always attaching low. Minipar always attaches low except for named entities of the form `NP Prep NP` (e.g. *The State Commission on Judicial Conduct*), which are recognized as a unit, resulting in high attachment for some *which* relative clauses. For *that* clauses, Minipar always attaches low.[5]

For *that* clauses we achieved results about 5 percentage points above the baseline; for *which* clauses about 5 to 7 points above the baseline, and about 3 points above Minipar.

set	not used	accuracy
development		64.42%
development	mod	64.16%
development	mod,f	63.90%
development	mod,f,obj	63.64%
development	mod,f,obj,c	63.38%
test		60.87%
test	mod	60.35%
test	mod,f	60.10%
test	mod,f,obj	60.10%
test	mod,f,obj,c	60.10%

Table 3: Accuracy on *that* clauses when the number of contextual features is decreased. The middle column shows what is left out (mod: the modifier is not used, f: only the head noun is used, obj: only the verb and not its object is used, c: the class/hypernym is not used.)

Tables 3 and 5 show how much of a decrease in accuracy is caused by using less context. For the development set the accuracy drops continuously as we omit an increasing number of elements of the context: pre-modifiers, lexical modifiers, objects, hypernyms. On the test set we can also observe a drop in accuracy. However, it is less consistent: Omitting the object does not decrease performance, and not using classes for named entities does have an effect on the *which* test set, but not on the *that* test set. These results show that using a larger context than just simple noun-verb co-occurrence improves performance and that a number of sources of information need to be combined for consistent improvement.

[4] We experimented with a number of values on our development set. Accuracy of the algorithm is only slightly affected for values between 4 and 7.

[5] Note that this property leads to a higher "Minipar" baseline for *which* clauses.

5.1 Integration into a statistical parser

After having shown the success of our method in a stand-alone evaluation, we now turn to evaluating it when integrated into a statistical parser, the Collins parser as reimplemented by (Bikel, 2004). We apply structural disambiguation (SD) to all *that*- and *which*-sentences of Sec. 13–24 with a relative clause attached to either the first or second NP in a pattern of the form "NP PREP NP RC". Sentences without an "NP PREP NP RC" structure in the gold standard are omitted (i.e., we don't attempt to correct spurious RC attachment ambiguity). Since we want to develop methods that can leverage small training sets, we perform the evaluation for 5 different training set sizes: 50%, 25%, 5%, 1%, and 0.1% of the Penn treebank, each a subset of Sec. 00–12 (Table 4). Note that the number of eligible relative clause constellations in the test set varies depending on the training set.

For *which* sentences, SD consistently improves parsing accuracy. For *that* sentences accuracy is improved for small training sets (0.1% and 1%). Differences that are significant according to the χ^2-test are indicated in the table. This demonstrates that our approach is successful especially in cases where the amount of training data available is limited.

Train data	# **which** sent.	Coll. only	Coll.+SD
50%	251	71.7%	78.5%
25%	250	70.0%	78.8%*
5%	238	68.9%	79.8%*
1%	245	67.8%	78.9%*
0.1%	194	60.8%	75.8%*
Train data	# **that** sent.	Coll. only	Coll.+SD
50%	366	72.7%	62.3%
25%	367	70.3%	61.9%
5%	356	67.4%	61.2%
1%	354	58.8%	60.2%
0.1%	314	47.5%	61.2%*

Table 4: Performance of the Collins parser (percent correct attachments) with and without structural disambiguation (SD). The combined method is superior for *which* and for small training sets. Significant improvements are marked with $*$.

6 Related Work

There have been few attempts to incorporate information from unlabeled corpora directly into the parser (Charniak, 1997; Johnson and Riezler, 2000), but they were either unsuccessful or tested on small data sets only. We know of no other work that combines attachment disambiguation based on unlabeled corpora with state-of-the-art statistical parsers.

Our lattice formalization can be viewed as a back-off model that combines estimates from several "backoffs" (in a typical back-off model, there is a single more general model to back off to). (Collins and Brooks, 1995) present a similar approach for prepositional phrases. One variant of their model computes the estimate in question as the average of three "backoffs." In contrast to prepositional phrases, many other attachment decisions, including relative clause attachments, are largely semantic. Given the verb *rely*, verb attachment of a PP headed by *on* is very likely. There are no similar strong regularities for semantic attachments: they require measuring the semantic "fit" of the two elements being syntactically attached to each other. This is why we use MI in this paper to disambiguate attachment. To our knowledge, MI has not been used in a back-off model before.

The lattice can also be viewed as a set of overlapping features, similar to the feature space of many discriminative algorithms. However, in contrast to discriminative learning, our approach is unsupervised.

There is a large body of literature on PP attachment, e.g. (Hindle and Rooth, 1991; Volk, 2001; Calvo et al., 2005) that shares the overall goals of this paper: using information from unlabeled corpora for syntactic disambiguation. (Volk, 2001) counts the number of occurrences of word n-grams on the web to select the correct attachment of PPs. We believe that grammatical dependencies are a more promising research direction since they are more robust compared to raw text if data are sparse. (Toutanova et al., 2004)'s approach is similar to ours in that morphological variants and word classes are considered, but their method differs in that they use both labelled corpora and unlabelled corpora for calculating attachment decisions. Work in the tradition of (Hindle and Rooth, 1991) is most similar to the approach proposed here. The authors parse an unannotated corpus and use dependency statistics for disambiguation of PP attachment. Our interest is in developing a framework that can disambiguate syntactic ambiguities in general, at least as far as they correspond to attachment ambiguities, as opposed to solving a particular syntactic ambiguity problem.

set	not used	accuracy
development		82.27%
development	mod	81.21%
development	mod,f	81.21%
development	mod,f,obj	80.49%
development	mod,f,obj,c	78.72%
test		78.41%
test	mod	78.41%
test	mod,f	78.03%
test	mod,f,obj	78.41%
test	mod,f,obj,c	78.03%

Table 5: Accuracy on *which* clauses, when the number of contextual features is decreased. (cf. Table 3 for further explanation.)

Previous work on relative clause attachment has taken a machine learning approach where an attachment decision is represented as a feature vector which is then fed into a classifier trained on a labeled training set. In contrast, our main emphasis is on exploiting information from unlabeled corpora. (Siddharthan, 2002a; Siddharthan, 2002b) uses WordNet classes for constructing some of the features characterizing attachments. For *which* clauses (Siddharthan, 2002b) achieves an accuracy of 76.5% on his test set.[6] RC attachment is also addressed by (Yeh and Vilain, 1998), who experiment with a transformation-based error-driven learning approach, which aims to disambiguate various cases of PP attachment ambiguities and subordinate clauses at the same time. They report an overall accuracy of 75.4%, but do not give numbers for relative clause attachment.

[6] We attempted to recreate Siddharthan's training and test sets, but were not able to based on the description in the paper and email communication with the author.

7 Conclusion

We make three contributions in this paper. First, we propose a lattice-based framework for combining supervised and unsupervised methods for syntactic disambiguation. Parses from a treebank-trained parser are refined by using additional information from a large unannotated corpus, represented as dependencies extracted by a dependency parser. The lattice integrates information obtained from variable context sizes. This approach makes it possible to base attachment decisions on the most specific context available in the unlabeled corpus.

Secondly, we evaluate attachment disambiguation by comparing to the performance of a state-of-the-art parser. Most previous work on attachment ambiguity has not been evaluated against this stringent baseline. We also argue that it is important to compare results across different training set sizes since in practical applications we can expect training sets to be smaller than is typical in academia.

Finally, we address the problem of relative clause attachment, a problem that has received much less attention than PP attachment. We argue that RC attachment is a good test case for enhancing statistical parsers with information from unlabeled corpora because it is more complex than PP attachment due to a wider range of grammatical phenomena involved and because few instances of RC attachment ambiguity can be resolved structurally. We also provide a baseline for future evaluations of work on RC attachment disambiguation.

References

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

Hiram Calvo, Alexander Gelbukh, and Adam Kilgarriff. 2005. Distributional thesaurus vs. wordnet: A comparison of backoff techniques for unsupervised PP attachment. In *CICLing 6*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL 43*.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *AAAI/IAAI*, pages 598–603.

Michael Collins and James Brooks. 1995. Prepositional attachment through a backed-off model. In David Yarovsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 27–38, Somerset, New Jersey. Association for Computational Linguistics.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70, March.

Donald Hindle and Mats Rooth. 1991. Structural ambiguity and lexical relations. In *Proc. of ACL 29*, pages 229–236, Morristown NJ. Association of Computational Linguistics.

Mark Johnson and Stefan Riezler. 2000. Exploiting auxiliary distributions in stochastic unification-based grammars. In *NAACL*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5.

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

LingPipe. 2006. http://www.alias-i.com/lingpipe/.

Lucene. 2006. http://lucene.apache.org.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large natural language corpus of English: the penn treebank. *Computational Linguistics*, 19:313–330.

Advaith Siddharthan. 2002a. Resolving attachment and clause boundary ambiguities for simplifying relative clause constructs. In *Student Research Workshop, ACL*.

Advaith Siddharthan. 2002b. Resolving relative clause attachment ambiguities using machine learning techniques and wordnet hierarchies. In *Proceedings of the 4th Discourse Anaphora and Anaphora Resolution Colloquium (DAARC 2002)*.

Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. 2004. Learning random walk models for inducing word dependency distributions. In *Proceedings of ICML*.

Martin Volk. 2001. Exploiting the WWW as a corpus to resolve pp attachment ambiguities. In *Proceedings of Corpus Linguistics 2001*.

Web Appendix. 2006. http://www.ims.uni-stuttgart.de/schuetze/conll06/apdx.html.

Alexander S. Yeh and Marc B. Vilain. 1998. Some properties of preposition and subordinate conjunction attachments. In *Coling 17*.

Word Distributions for Thematic Segmentation in a Support Vector Machine Approach

Maria Georgescul
ISSCO/TIM, ETI
University of Geneva
1211 Geneva, Switzerland
maria.georgescul@eti.unige.ch

Alexander Clark
Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK
alexc@cs.rhul.ac.uk

Susan Armstrong
ISSCO/TIM, ETI
University of Geneva
1211 Geneva, Switzerland
susan.armstrong@issco.unige.ch

Abstract

We investigate the appropriateness of using a technique based on support vector machines for identifying thematic structure of text streams. The thematic segmentation task is modeled as a binary-classification problem, where the different classes correspond to the presence or the absence of a thematic boundary. Experiments are conducted with this approach by using features based on word distributions through text. We provide empirical evidence that our approach is robust, by showing good performance on three different data sets. In particular, substantial improvement is obtained over previously published results of word-distribution based systems when evaluation is done on a corpus of recorded and transcribed multi-party dialogs.

1 Introduction

(Todd, 2005) distinguishes between "local-level topics (of sentences, utterances and short discourse segments)" and "discourse topics (of more extended stretches of discourse)".[1] (Todd, 2005) points out that "discourse-level topics are one of the most elusive and intractable notions in semantics". Despite this difficulty in giving a rigorous definition of *discourse topic*, the task of discourse/dialogue segmentation into thematic episodes can be described by invoking an "intuitive notion of topic" (Brown and Yule, 1998). Thematic segmentation also relates to several notions such as speaker's intention, topic flow and cohesion.

In order to find out if thematic segment identification is a feasible task, previous state-of-the-art works appeal to experiments, in which several human subjects are asked to mark thematic segment boundaries based on their intuition and a minimal set of instructions. In this manner, previous studies, e.g. (Passonneau and Litman, 1993; Galley et al., 2003), obtained a level of inter-annotator agreement that is statistically significant.

Automatic thematic segmentation (TS), i.e. the segmentation of a text stream into topically coherent segments, is an important component in applications dealing with large document collections such as information retrieval and document browsing. Other tasks that could benefit from the thematic textual structure include anaphora resolution, automatic summarisation and discourse understanding.

The work presented here tackles the problem of TS by adopting a supervised learning approach for capturing linear document structure of non-overlapping thematic episodes. A prerequisite for the input data to our system is that texts are divided into sentences or utterances.[2] Each boundary between two consecutive utterances is a potential thematic segmentation point and therefore, we model the TS task as a binary-classification problem, where each utterance should be classified as marking the

[1] In this paper, we make use of the term *topic* or *theme* as referring to the discourse/dialogue topic.

[2] Occasionally within this document we employ the term utterance to denote either a sentence or an utterance in its proper sense.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 101–108, New York City, June 2006.

presence or the absence of a topic shift in the discourse/dialogue based only on observations of patterns in vocabulary use.

The remainder of the paper is organised as follows. The next section summarizes previous techniques, describes how our method relates to them and presents the motivations for a support vector approach. Sections 3 and 4 present our approach in adopting support vector learning for thematic segmentation. Section 5 outlines the empirical methodology and describes the data used in this study. Section 6 presents and discusses the evaluation results. The paper closes with Section 7, which briefly summarizes this work and offers some conclusions and future directions.

2 Related Work

As in many existing approaches to the thematic segmentation task, we make the assumption that the thematic coherence of a text segment is reflected at lexical level and therefore we attempt to detect the correlation between word distribution and thematic changes throughout the text. In this manner, (Hearst, 1997; Reynar, 1998; Choi, 2000) start by using a similarity measure between sentences or fixed-size blocks of text, based on their word frequencies in order to find changes in vocabulary use and therefore the points at which the topic changes. Sentences are then grouped together by using a clustering algorithm. (Utiyama and Isahara, 2001) models the problem of TS as a problem of finding the minimum cost path in a graph and therefore adopts a dynamic programming algorithm. The main advantage of such methods is that no training time and corpora are required.

By modeling TS as binary-classification problem, we introduce a new technique based on support vector machines (SVMs). The main advantage offered by SVMs with respect to methods such as those described above is related to the distance (or similarity) function used. Thus, although (Choi, 2000; Hearst, 1997) employ a distance function (i.e. *cosine distance*) to detect thematic shifts, SVMs are capable of using a larger variety of similarity functions.

Moreover, SVMs can employ distance functions that operate in extremely high dimensional feature spaces. This is an important property for our task, where handling high dimensionality data representation is necessary (see section 4).

An alternative to dealing with high dimension data may be to reduce the dimensionality of the data representation. Therefore, linear algebra dimensionality reduction methods like singular value decomposition have been adopted by (Choi et al., 2001; Popescu-Belis et al., 2004) in Latent Semantic Analysis (LSA) for the task of thematic segmentation. A Probabilistic Latent Semantic Analysis (PLSA) approach has been adopted by (Brants et al., 2002; Farahat and Chen, 2006) for the TS task. (Blei and Moreno, 2001) proposed a TS approach, by embedding a PLSA model in an extended Hidden Markov Model (HMM) approach, while (Yamron et al., 1998) have previously proposed a HMM approach for TS.

A shortcoming of the methods described above is due to their typically generative manner of training, i.e. using the maximum likelihood estimation for a joint sampling model of observation and label sequences. This poses the challenge of finding more appropriate *objective functions*, i.e. alternatives to the log-likelihood that are more closely related to application-relevant performance measures. Secondly, efficient inference and learning for the TS task often requires making questionable conditional independence assumptions. In such cases, improved performance may be obtained by using methods with a more discriminative character, by allowing direct dependencies between a label and past/future observations and by efficient handling higher-order combinations of input features. Given the discriminative character of SVMs, we expect our model to attain similar benefits.

3 Support Vector Learning Task and Thematic Segmentation

The theory of Vapnik and Chervonenkis (Vapnik, 1995) motivated the introduction of support vector learning. SVMs have originally been used for classification purposes and their principles have been extended to the task of regression, clustering and feature selection. (Kauchak and Chen, 2005) employed SVMs using features (derived for instance from information given by the presence of paragraphs, pronouns, numbers) that can be reliably used for topic

segmentation of narrative documents. Aside from the fact that we consider the TS task on different datasets (not only on narrative documents), our approach is different from the approach proposed by (Kauchak and Chen, 2005) mainly by the data representation we propose and by the fact that we put the emphasis on deriving the thematic structure merely from word distribution, while (Kauchak and Chen, 2005) observed that the 'block similarities provide little information about the actual segment boundaries' on their data and therefore they concentrated on exploiting other features.

An excellent general introduction to SVMs and other kernel methods is given for instance in (Cristianini and Shawe-Taylor, 2000). In the section below, we give some highlights representing the main elements in using SVMs for thematic segmentation.

The support vector learner $\mathcal{L}$ is given a *training set* of n *examples*, usually denoted by S_{train}= (($\vec{u}_1$, y_1),...,($\vec{u}_n$, y_n))$\subseteq$ $(U \times Y)^n$ drawn independently and identically distributed according to a fixed distribution $Pr(u, y) = Pr(y|u)Pr(u)$. Each training example consists of a high-dimensional vector $\vec{u}$ describing an utterance and the class label y. The utterance representations we chose are further described in Section 4. The class label y has only two possible values: 'thematic boundary' or 'non-thematic boundary'. For notational convenience, we replace these values by +1 and -1 respectively, and thus we have $y \in$ {-1, 1}. Given a hypothesis space $\mathcal{H}$, of functions $h : U \rightarrow \{-1, +1\}$ having the form $h(\vec{u}) = sign(< \vec{w}, \vec{u} > +b)$, the inductive support vector learner $\mathcal{L}_{ind}$ seeks a decision function h_{ind} from $\mathcal{H}$, using S_{train} so that the expected number of erroneous predictions is minimized. Using the structural risk minimization principle (Vapnik, 1995), the support vector learner gets the optimal decision function h by minimizing the following cost function:

$$\mathcal{W}^{ind}(\vec{w}, b, \xi_1, \xi_2, ..., \xi_n) = \tfrac{1}{2} < \vec{w}, \vec{w} > +$$

$$+ C^+ \sum_{i=0,y_i=1}^{n} \xi_i + C^- \sum_{i=0,y_i=-1}^{n} \xi_i,$$

subject to:

$$y_i[< \vec{w} \cdot \vec{u}_i > +b] \ \leq 1 - \xi_i \ for \ i = 1, 2, ..., n;$$

$$\xi_i \geq 0 \ for \ i = 1, 2, ..., n.$$

The parameters $\vec{w}$ and b follow from the optimisation problem, which is solved by applying Lagrangian theory. The so-called *slack variables* ξ_i, are introduced in order to be able to handle non-separable data. The positive parameters C^+ and C^- are called *regularization parameters* and determine the amount up to which errors are tolerated. More exactly, training data may contain noisy or outlier data that are not representative of the underlying distribution. On the one hand, fitting exactly to the training data may lead to overfitting. On the other hand, dismissing true properties of the data as sampling bias in the training data will result in low accuracy. Therefore, the regularization parameter is used to balance the trade-off between these two competing considerations. Setting the regularization parameter too low can result in poor accuracy, while setting it too high can lead to overfitting. In the TS task, we used an automated procedure to select the regularization parameters, as further described in section 5.3.

In cases where non-linear hypothesis functions should be optimised, each $\vec{u}_i$ can be mapped into $\varphi(\vec{u}_i) \in F$, where F is a higher dimensional space usually called *feature space*, in order to make linear the relation between $\vec{u}_i$ and y_i. Thus the original linear learning machine can be adopted in finding the classification solution in the feature space.

When using a mapping function $\varphi : U \rightarrow F$, if we have a way of computing the inner product $\langle \varphi(\vec{u}_i), \varphi(\vec{u}_j) \rangle$ directly as a function of the original input point, then the so-called kernel function $K(\vec{u}_i, \vec{u}_j) = \langle \varphi(\vec{u}_i), \varphi(\vec{u}_j) \rangle$ is proved to simplify the computational complexity implied by the direct use of the mapping function φ. The choice of appropriate kernels and its specific parameters is an empirical issue. In our experiments, we used the Gaussian radial basis function (RBF) kernel:

$$K_{RBF}(\vec{u}_i, \vec{u}_j) = \exp(-\gamma^2 ||\vec{u}_i - \vec{u}_j||^2).$$

For the SVM calculations, we used the *LIBSVM* library (Chang and Lin, 2001).

4 Representation of the information used to determine thematic boundaries

As presented in section 3, in the thematic segmentation task, an input $\vec{u}_i$ to the support vector classifier is a vectorial representation of the utterance to

be classified and its context. Each dimension of the input vector indicates the value of a certain feature characterizing the utterance. All input features here are indicator functions for a word occurring within a fixed-size window centered on the utterance being labeled. More exactly, the input features are computed in the following steps:

1. The text has been pre-processed by tokenization, elimination of stop-words and lemmatization, using *TreeTagger* (Schmid, 1996).

2. We make use of the so-called *bag of words* approach, by mapping each utterance to a *bag*, i.e. a set that contains word frequencies. Therefore, word frequencies have been computed to count the number of times that each term (i.e. word lemma) is used in each utterance. Then a transformation of the raw word frequency counts is applied in order to take into account both the local (i.e. for each utterance) word frequencies as well as the overall frequencies of their occurrences in the entire text collection. More exactly, we made experiments in parallel with three such transformations, which are very commonly used in information retrieval domain (Dumais, 1991): *tf.idf*, *tf.normal* and *log.entropy*.

3. Each i-th utterance is represented by a vector $\vec{u}_i$, where a j-th element of $\vec{u}_i$ is computed as:

$$u_{i,j} = \left(\sum_{t=i-winSize}^{i} f_{t,j} \right) \left(\sum_{k=i+1}^{i+winSize} f_{k,j} \right),$$

   where $winSize \geq 1$ and $f_{i,j}$ is the weighted frequency (determined in the previous step) of the j-th word from the vocabulary in the i-th utterance. In this manner, we will have $u_{i,j} > 0$ if and only if at least two occurrences of the j-th term occur within $(2 \cdot winSize)$ utterances on opposite sides of a boundary candidate. That is, each $u_{i,j}$ is capturing how many word co-occurrences appear across the candidate utterance in an interval (of $(2 \cdot winSize)$ utterances) centered in the boundary candidate utterance.

4. Each attribute value from the input data is scaled to the interval $[0, 1]$.

Note that the vector space representation adopted in the previous steps will result in a sparse high dimensional input data for our system. More exactly, table 1 shows the average number of non-zero features per example corresponding to each data set (further described in section 5.1).

Data set	Non zero features
ICSI	3.67%
TDT	0.40%
Brown	0.12%

Table 1: The percentage of non-zero features per example.

5 Experimental Setup

5.1 Data sets used

In order to evaluate how robust our SVM approach is, we performed experiments on three English data sets of approximately the same dimension (i.e. containing about 260,000 words).

The first dataset is a subset of the ICSI-MR corpus (Janin et al., 2004), where the gold standard for thematic segmentations has been provided by taking into account the agreement of at least three human annotators (Galley et al., 2003). The corpus consists of high-quality close talking microphone recordings of multi-party dialogues. Transcriptions at word level with utterance-level segmentations are also available. A test sample from this dataset consists of the transcription of an approximately one-hour long meeting and contains an average of about seven thematic episodes.

The second data set contains documents randomly selected from the Topic Detection and Tracking (TDT) 2 collection, made available by (LDC, 2006). The TDT collection includes broadcast news and newswire text, which are segmented into topically cohesive stories. We use the story segmentation provided with the corpus as our gold standard labeling. A test sample from our subset contains an average of about 24 segments.

The third dataset we use in this study was originally proposed in (Choi, 2000) and contains artificial thematic episodes. More precisely, the dataset is built by concatenating short pieces of texts that

Data set	Weighting schema	winSize	γ	*C*
ICSI	log.entropy	57	0.0625	0.01
TDT	tf.idf	17	0.0625	0.1
Brown	tf.idf	5	0.0625	0.001

Table 2: The optimal settings found for the SVM model, using the RBF kernel.

have been randomly extracted from the Brown corpus. Any test sample from this dataset consists of ten segments. Each segment contains at least three sentences and no more than eleven sentences.

While the focus of our paper is not on the method of evaluation, it is worth pointing out that the performance on the synthetic data set is a very poor guide to the performance on naturally occurring data (Georgescul et al., 2006). We include the synthetic data for comparison purposes.

5.2 Handling unbalanced data

We have a small percentage of positive examples relative to the total number of training examples. Therefore, in order to ensure that positive points are not considered as being noisy labels, we change the penalty of the minority (positive) class by setting the parameter C^+ of this class to:

$$C^+ = \lambda \cdot \left(\frac{n}{n^+ - 1} - 1 \right) \cdot C^-,$$

where n^+ is the number of positive training examples, n is the total number of training examples and λ is the scaling factor. In the experiments reported here, we set the value for the scale factor λ to $\lambda = 1$ and we have: $C^+ = 7 \cdot C^-$ for the synthetic data derived from Brown corpus; $C^+ = 18 \cdot C^-$for the TDT data and $C^+ = 62 \cdot C^-$ for the ICSI meeting data.

5.3 Model selection

We used 80% of each dataset to determine the best model settings, while the remaining 20% is used for testing purposes. Each training set (for each dataset employed) was divided into disjoint subsets and five-fold cross-validation was applied for model selection.

In order to avoid too many combinations of parameter settings, model selection is done in two phases, by distinguishing two kinds of parameters. First, the parameters involved in data representation (see section 4) are addressed. We start with choosing an appropriate term weighting scheme and a good value for the $winSize$ parameter. This choice is based on a systematic grid search over 20 different values for $winSize$ and the three variants *tf.idf*, *tf.normal* and *log.entropy* for term weighting. We ran five-fold cross validation, by using the RBF kernel with its parameter γ fixed to $\gamma = 1$. We also set the regularization parameter C equal to C = 1.

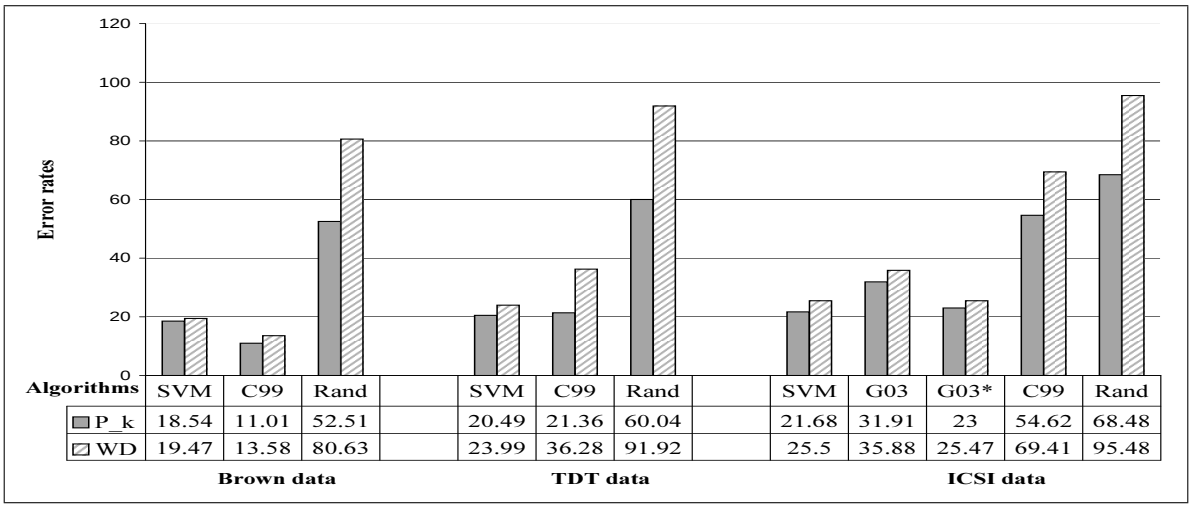
In the second phase of model selection, we take the optimal parameter values selected in the previous phase as a constant factor and search the most appropriate values for C and γ parameters. The range of values we select from is: $C \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$ and $\gamma \in \{2^{-6}, 2^{-5}, 2^{-4}, ..., 2^4, 2^6\}$ and for each possible value we perform five-fold cross validation. Therefore, we ran the algorithm five times for the $91 = 7 \times 13$ parameter settings. The most suitable model settings found are shown in Table 2. For these settings, we show the algorithm's results in section 6.

6 Evaluation

6.1 Evaluation Measures

Beeferman et al. (1999) underlined that the standard evaluation metrics of *precision* and *recall* are inadequate for thematic segmentation, namely by the fact that these metrics did not account for how far away a hypothesized boundary (i.e. a boundary found by the automatic procedure) is from the reference boundary. On the other hand, for instance, an algorithm that places a boundary just one utterance away from the reference boundary should be penalized less than an algorithm that places a boundary ten (or more) utterances away from the reference boundary.

Hence the use of two other evaluation metrics is favored in thematic segmentation: the P_k metric (Beeferman et al., 1999) and the *WindowDiff* error metric (Pevzner and Hearst, 2002). In con-



Algorithms	SVM	C99	Rand		SVM	C99	Rand		SVM	G03	G03*	C99	Rand
P_k	18.54	11.01	52.51		20.49	21.36	60.04		21.68	31.91	23	54.62	68.48
WD	19.47	13.58	80.63		23.99	36.28	91.92		25.5	35.88	25.47	69.41	95.48
	Brown data				TDT data				ICSI data				

Figure 1: Error rates of the segmentation systems.

trast to precision and recall, these metrics allow for a slight vagueness in where the hypothesized thematic boundaries are placed and capture "the notion of nearness in a principled way, gently penalizing algorithms that hypothesize boundaries that aren't quite right, and scaling down with the algorithm's degradation" (Beeferman et al., 1999). That is, computing both P_k and *WindowDiff* metrics involves the use of a fixed-size (i.e. having a fixed number of either words or utterances) window that is moved step by step over the data. At each step, P_k and *WindowDiff* are basically increased (each metric in a slightly different way) if the hypothesized boundaries and the reference boundaries are not within the same window.

During the model selection phase, we used precision and recall in order to measure the system's error rate. This was motivated by the fact that posing the TS task as a classification problem leads to a loss of the sequential nature of the data, which is an inconvenient in computing the P_k and *WindowDiff* measures. However, during the final testing phase of our system, as well as for the evaluation of the previous systems, we use both the P_k and the *WindowDiff* error metric.

The relatively small size of our datasets does not allow for dividing our test set into multiple sub-test sets for applying statistical significance tests. This would be desirable in order to indicate whether the differences in system error rates are statistically significant over different data sets. Nevertheless, we believe that measuring differences in error rates obtained on the test set is indicative of the relative performance. Thus, the experimental results shown in this paper should be considered as illustrative rather than exhaustive.

6.2 Results

In order to determine the adequacy of our SVM approach over different genres, we ran our system over three datasets, namely the ICSI meeting data, the TDT broadcast data and the Brown written genre data.

By measuring the system error rates using the P_k and the *WindowDiff* metrics, Figure 1 summarizes the quantitative results obtained in our empirical evaluation. In Figure 1, our SVM approach is labeled as *SVM* and we abbreviate *WindowDiff* as *WD*. The results of our *SVM* system correspond to the parameter values detected during model selection (see Table 2). We compare our system against an existing thematic segmenter in the literature: *C99* (Choi, 2000). We also give for comparison the error rates of a naive algorithm, labeled as *Rand* algorithm, which randomly distributes boundaries throughout the text.

The *LCseg* system (Galley et al., 2003), labeled here as *G03*, is to our knowledge the only word distribution based system evaluated on ICSI meeting data. Therefore, we replicate the results reported by (Galley et al., 2003) when evaluation of *LCseg* was done on ICSI data. The so-labeled *G03** algorithm

indicates the error rates obtained by (Galley et al., 2003) when extra (meeting specific) features have been adopted in a decision tree classifier. However, note that the results reported by (Galley et al.) are not directly comparable with our results because of a slight difference in the evaluation procedure: (Galley et al.) performed 25-fold cross validation and the average P_k and WD error rates have been computed on the held-out sets.

Figure 1 illustrates the following interesting results. For the ICSI meeting data, our SVM approach provides the best performance relative to the competing word distribution based state-of-the-art methods. This proves that our SVM-based system is able to build a parametric model that leads to a segmentation that highly correlates to a human thematic segmentation. Furthermore, by taking into account the relatively small size of the data set we used for training, it can be concluded that the SVM can build qualitatively good models even with a small training data. The work of (Galley et al., 2003) shows that the *G03** algorithm is better than *G03* by approximately 10%, which indicates that on meeting data the performance of our word-distribution based approach could possibly be increased by using other meeting-specific features.

By examining the error rates given by P_k metric for the three systems on the TDT data set, we observe that our system and *C99* performed more or less equally. With respect to the *WindowDiff* metric, our system has an error rate approximately 10% smaller than *C99*.

On the synthetic data set, the *SVM* approach performed slightly worse than *C99*, avoiding however catastrophic failure, as observed with the *C99* method on ICSI data.

7 Conclusions

We have introduced a new approach based on word distributions for performing thematic segmentation. The thematic segmentation task is modeled here as a binary classification problem and support vector machine learning is adopted. In our experiments, we make a comparison of our approach versus existing linear thematic segmentation systems reported in the literature, by running them over three different data sets. When evaluating on real data, our approach either outperformed the other existing methods or performs comparably to the best. We view this as a strong evidence that our approach provides a unified and robust framework for the thematic segmentation task. The results also suggest that word distributions themselves might be a good candidate for capturing the thematic shifts of text and that SVM learning can play an important role in building an adaptable correlation.

Our experiments also show the sensitivity of a segmentation method to the type of a corpus on which it is tested. For instance, the C99 algorithm which achieves superior performance on a synthetic collection performs quite poorly on the real-life data sets.

While we have shown empirically that our technique can provide considerable gains by using single word distribution features, future work will investigate whether the system can be improved by exploiting other features derived for instance from syntactic, lexical and, when available, prosodic information. If further annotated meeting data becomes available, it would be also interesting to replicate our experiments on a bigger data set in order to verify whether our system performance improves.

Acknowledgments This work is partially supported by the Interactive Multimodal Information Management project (http://www.im2.ch/). Many thanks to the reviewers for their insightful suggestions. We are grateful to the International Computer Science Institute (ICSI), University of California for sharing the data with us. The authors also thank Michael Galley who kindly provided us the thematic annotations of the ICSI data.

References

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical Models for Text Segmentation. *Machine Learning*, 34(1-3):177–210.

David M. Blei and Pedro J. Moreno. 2001. Topic Segmentation with an Aspect Hidden Markov Model. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 343–348. ACM Press.

Thorsten Brants, Francine Chen, and Ioannis Tsochantaridis. 2002. Topic-Based Document Segmentation with Probabilistic Latent Semantic Analysis. In *Proceedings of the Eleventh International Conference on*

Information and Knowledge Management, pages 211–218, McLean, Virginia, USA. ACM Press.

Gillian Brown and George Yule. 1998. *Discourse Analysis*. Cambridge Textbooks in Linguistics, Cambridge.

Chih-Chung Chang and Chih-Jen Lin. 2001. LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Freddy Choi, Peter Wiemer-Hastings, and Johanna Moore. 2001. Latent Semantic Analysis for Text Segmentation. In *Proceedings of the 6th Conference on Empirical Methods in Natural Language Processing*, Seattle, WA.

Freddy Choi. 2000. Advances in Domain Independent Linear Text Segmentation. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, pages 26–33, Seattle, USA.

Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, UK.

Susan Dumais. 1991. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 23(2):229–236.

Ayman Farahat and Francine Chen. 2006. Improving Probabilistic Latent Semantic Analysis with Principal Component Analysis. In *Proceedings of the 11th Conference of the European Chapter of the Asociation for Computational Linguistics*, Trento, Italy.

Michael Galley, Kathleen McKeown, Eric Fosler-Luissier, and Hongyan Jing. 2003. Discourse Segmentation of Multy-Party Conversation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 562–569.

Maria Georgescul, Alexander Clark, and Susan Armstrong. 2006. An Analysis of Quantitative Aspects in the Evaluation of Thematic Segmentation Algorithms. *To appear*.

Marti Hearst. 1997. TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages. *Computational Linguistics*, 23(1):33–64.

Adam Janin, Jeremy Ang, Sonali Bhagat, Rajdip Dhillon, Jane Edwards, Javier Macias-Guarasa, Nelson Morgan, Barbara Peskin, Elizabeth Shriberg, Andreas Stolcke, Chuck Wooters, and Britta Wrede. 2004. The ICSI Meeting Project: Resources and Research. In *ICASSP 2004 Meeting Recognition Workshop (NIST RT-04 Spring Recognition Evaluation)*, Montreal.

David Kauchak and Francine Chen. 2005. Feature-Based Segmentation of Narrative Documents. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, pages 32–39, Ann Arbor; MI; USA.

LDC. 2006. The Linguistic Data Consortium. Available from World Wide Web: http://www.ldc.upenn.edu.

Rebecca J. Passonneau and Diane J. Litman. 1993. Intention-based Segmentation: Human Reliability and Correlation with Linguistic Cues. In *Proceedings of the 31st conference on Association for Computational Linguistics*, pages 148 – 155, Columbus, Ohio.

Lev Pevzner and Marti Hearst. 2002. A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Computational Linguistics*, 16(1):19–36.

Andrei Popescu-Belis, Alexander Clark, Maria Georgescul, Sandrine Zufferey, and Denis Lalanne. 2004. Shallow Dialogue Processing Using Machine Learning Algorithms (or Not). In Bourlard H. and Bengio S., editors, *Multimodal Interaction and Related Machine Learning Algorithms*, pages 277–290. LNCS 3361, Springer-Verlag, Berlin.

Jeffrey Reynar. 1998. *Topic Segmentation: Algorithms and Applications*. Ph.D. thesis, University of Pennsylvania.

Helmut Schmid. 1996. Probabilistic Part-of-Speech Tagging Using Decision Trees. Technical report, Institute for Computational Linguistics of the University of Stuttgart.

Richard Watson Todd. 2005. A fuzzy approach to discourse topics. *Journal of the International Association for Semiotic Studies*, 155:93–123.

Masao Utiyama and Hitoshi Isahara. 2001. A Statistical Model for Domain-Independent Text Segmentation. In *Proceedings of the 39th Annual Meeting of the ACL joint with the 10th Meeting of the European Chapter of the ACL*, pages 491–498, Toulouse, France.

Vladimir Naumovich Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Jonathan P. Yamron, Ira Carp, Lawrence Gillick, Stewe Lowe, and Paul van Mulbregt. 1998. A Hidden Markov Model Approach to Text Segmentation and Event Tracking. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, volume 17, pages 333–336, Seattle, WA.

Which Side are You on? Identifying Perspectives at the Document and Sentence Levels

Wei-Hao Lin
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
whlin@cs.cmu.edu

Theresa Wilson, Janyce Wiebe
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260
{twilson,wiebe}@cs.pitt.edu

Alexander Hauptmann
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
alex@cs.cmu.edu

Abstract

In this paper we investigate a new problem of identifying the *perspective* from which a document is written. By perspective we mean a point of view, for example, from the perspective of Democrats or Republicans. Can computers learn to identify the perspective of a document? Not every sentence is written strongly from a perspective. Can computers learn to identify which sentences strongly convey a particular perspective? We develop statistical models to capture how perspectives are expressed at the document and sentence levels, and evaluate the proposed models on articles about the Israeli-Palestinian conflict. The results show that the proposed models successfully learn how perspectives are reflected in word usage and can identify the perspective of a document with high accuracy.

1 Introduction

In this paper we investigate a new problem of automatically identifying the *perspective* from which a document is written. By perspective we mean a "subjective evaluation of relative significance, a point-of-view."[1] For example, documents about the Palestinian-Israeli conflict may appear to be about the same topic but reveal different perspectives:

(1) The inadvertent killing by Israeli forces of Palestinian civilians – usually in the course of shooting at Palestinian terrorists – is considered no different at the moral and ethical level than the deliberate targeting of Israeli civilians by Palestinian suicide bombers.

(2) In the first weeks of the Intifada, for example, Palestinian public protests and civilian demonstrations were answered brutally by Israel, which killed tens of unarmed protesters.

Example 1 is written from an Israeli perspective; Example 2 is written from a Palestinian perspective. Anyone knowledgeable about the issues of the Israeli-Palestinian conflict can easily identify the perspectives from which the above examples were written. However, can computers learn to identify the perspective of a document given a training corpus?

When an issue is discussed from different perspectives, not every sentence strongly reflects the perspective of the author. For example, the following sentences were written by a Palestinian and an Israeli.

(3) The Rhodes agreements of 1949 set them as the ceasefire lines between Israel and the Arab states.

(4) The green line was drawn up at the Rhodes Armistice talks in 1948-49.

Examples 3 and 4 both factually introduce the background of the issue of the "green line" without expressing explicit perspectives. Can we develop a

[1] The American Heritage Dictionary of the English Language, 4th ed.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X), pages 109–116, New York City, June 2006.

system to automatically discriminate between sentences that strongly indicate a perspective and sentences that only reflect shared background information?

A system that can automatically identify the perspective from which a document is written will be a valuable tool for people analyzing huge collections of documents from different perspectives. Political analysts regularly monitor the positions that countries take on international and domestic issues. Media analysts frequently survey broadcast news, newspapers, and weblogs for differing viewpoints. Without the assistance of computers, analysts have no choice but to read each document in order to identify those from a perspective of interest, which is extremely time-consuming. What these analysts need is to find strong statements from different perspectives and to ignore statements that reflect little or no perspective.

In this paper we approach the problem of learning individual perspectives in a statistical framework. We develop statistical models to learn how perspectives are reflected in word usage, and we treat the problem of identifying perspectives as a classification task. Although our corpus contains document-level perspective annotations, it lacks sentence-level annotations, creating a challenge for learning the perspective of sentences. We propose a novel statistical model to overcome this problem. The experimental results show that the proposed statistical models can successfully identify the perspective from which a document is written with high accuracy.

2 Related Work

Identifying the perspective from which a document is written is a subtask in the growing area of automatic opinion recognition and extraction. Subjective language is used to express opinions, emotions, and sentiments. So far, research in automatic opinion recognition has primarily addressed learning subjective language (Wiebe et al., 2004; Riloff et al., 2003), identifying opinionated documents (Yu and Hatzivassiloglou, 2003) and sentences (Yu and Hatzivassiloglou, 2003; Riloff et al., 2003), and discriminating between positive and negative language (Pang et al., 2002; Morinaga et al., 2002; Yu and Hatzivassiloglou, 2003; Turney and Littman, 2003; Dave et al., 2003; Nasukawa and Yi, 2003; Popescu and Etzioni, 2005; Wilson et al., 2005). While by its very nature we expect much of the language that is used when presenting a perspective or point-of-view to be subjective, labeling a document or a sentence as subjective is not enough to identify the perspective from which it is written. Moreover, the ideology and beliefs authors possess are often expressed in ways other than positive or negative language toward specific targets.

Research on the automatic classification of movie or product reviews as positive or negative (e.g., (Pang et al., 2002; Morinaga et al., 2002; Turney and Littman, 2003; Nasukawa and Yi, 2003; Mullen and Collier, 2004; Beineke et al., 2004; Hu and Liu, 2004)) is perhaps the most similar to our work. As with review classification, we treat perspective identification as a document-level classification task, discriminating, in a sense, between different types of opinions. However, there is a key difference. A positive or negative opinion toward a particular movie or product is fundamentally different from an overall perspective. One's opinion will change from movie to movie, whereas one's perspective can be seen as more static, often underpinned by one's ideology or beliefs about the world.

There has been research in discourse analysis that examines how different perspectives are expressed in political discourse (van Dijk, 1988; Pan et al., 1999; Geis, 1987). Although their research may have some similar goals, they do not take a computational approach to analyzing large collections of documents. To the best of our knowledge, our approach to automatically identifying perspectives in discourse is unique.

3 Corpus

Our corpus consists of articles published on the `bitterlemons` website[2]. The website is set up to "contribute to mutual understanding [between Palestinians and Israelis] through the open exchange of ideas."[3] Every week an issue about the Israeli-Palestinian conflict is selected for discussion (e.g.,

[2] `http://www.bitterlemons.org`

[3] `http://www.bitterlemons.org/about/about.html`

"Disengagement: unilateral or coordinated?"), and a Palestinian editor and an Israeli editor each contribute one article addressing the issue. In addition, the Israeli and Palestinian editors invite one Israeli and one Palestinian to express their views on the issue (sometimes in the form of an interview), resulting in a total of four articles in a weekly edition. We choose the `bitterlemons` website for two reasons. First, each article is already labeled as either Palestinian or Israeli by the editors, allowing us to exploit existing annotations. Second, the `bitterlemons` corpus enables us to test the generalizability of the proposed models in a very realistic setting: training on articles written by a small number of writers (two editors) and testing on articles from a much larger group of writers (more than 200 different guests).

We collected a total of 594 articles published on the website from late 2001 to early 2005. The distribution of documents and sentences are listed in Table 1. We removed metadata from all articles, including edition numbers, publication dates, topics, titles, author names and biographic information. We used OpenNLP Tools[4] to automatically extract sentence boundaries, and reduced word variants using the Porter stemming algorithm.

	Palestinian	Israeli
Written by editors	148	149
Written by guests	149	148
Total number of documents	297	297
Average document length	740.4	816.1
Number of sentences	8963	9640

Table 1: The basic statistics of the corpus

We evaluated the subjectivity of each sentence using the automatic subjective sentence classifier from (Riloff and Wiebe, 2003), and find that 65.6% of Palestinian sentences and 66.2% of Israeli sentences are classified as subjective. The high but almost equivalent percentages of subjective sentences in the two perspectives support our observation in Section 2 that a perspective is largely expressed using subjective language, but that the amount of subjectivity in a document is not necessarily indicative of its perspective.

[4] `http://sourceforge.net/projects/opennlp/`

4 Statistical Modeling of Perspectives

We develop algorithms for learning perspectives using a statistical framework. Denote a training corpus as a set of documents W_n and their perspectives labels $D_n, n = 1, \ldots, N$, where N is the total number of documents in the corpus. Given a new document $\tilde{W}$ with a unknown document perspective, the perspective $\tilde{D}$ is calculated based on the following conditional probability.

$$P(\tilde{D}|\tilde{W}, \{D_n, W_n\}_{n=1}^{N}) \quad (5)$$

We are also interested in how strongly each sentence in a document conveys perspective information. Denote the intensity of the m-th sentence of the n-th document as a binary random variable $S_{m,n}$. To evaluate $S_{m,n}$, how strongly a sentence reflects a particular perspective, we calculate the following conditional probability.

$$P(S_{m,n}|\{D_n, W_n\}_{n=1}^{N}) \quad (6)$$

4.1 Naïve Bayes Model

We model the process of generating documents from a particular perspective as follows:

$$\begin{aligned} \pi &\sim \text{Beta}(\alpha_\pi, \beta_\pi) \\ \theta &\sim \text{Dirichlet}(\alpha_\theta) \\ D_n &\sim \text{Binomial}(1, \pi) \\ W_n &\sim \text{Multinomial}(L_n, \theta_d) \end{aligned}$$

First, the parameters π and θ are sampled once from prior distributions for the whole corpus. Beta and Dirichlet are chosen because they are conjugate priors for binomial and multinomial distributions, respectively. We set the hyperparameters α_π, β_π, and α_θ to one, resulting in non-informative priors. A document perspective D_n is then sampled from a binomial distribution with the parameter π. The value of D_n is either d^0 (Israeli) or d^1 (Palestinian). Words in the document are then sampled from a multinomial distribution, where L_n is the length of the document. A graphical representation of the model is shown in Figure 1.

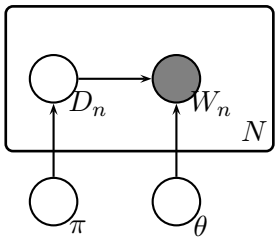


Figure 1: Naïve Bayes Model

The model described above is commonly known as a naïve Bayes (NB) model. NB models have been widely used for various classification tasks, including text categorization (Lewis, 1998). The NB model is also a building block for the model described later that incorporates sentence-level perspective information.

To predict the perspective of an unseen document using naïve Bayes , we calculate the posterior distribution of $\tilde{D}$ in (5) by integrating out the parameters,

$$\int\int P(\tilde{D}, \pi, \theta | \{(D_n, W_n)\}_{n=1}^{N}, \tilde{W}) d\pi d\theta \quad (7)$$

However, the above integral is difficult to compute. As an alternative, we use Markov Chain Monte Carlo (MCMC) methods to obtain samples from the posterior distribution. Details about MCMC methods can be found in Appendix A.

4.2 Latent Sentence Perspective Model

We introduce a new binary random variable, S, to model how strongly a perspective is reflected at the sentence level. The value of S is either s^1 or s^0, where s^1 indicates a sentence is written strongly from a perspective while s^0 indicates it is not. The whole generative process is modeled as follows:

$$\begin{aligned}
\pi &\sim \text{Beta}(\alpha_\pi, \beta_\pi) \\
\tau &\sim \text{Beta}(\alpha_\tau, \beta_\tau) \\
\theta &\sim \text{Dirichlet}(\alpha_\theta) \\
D_n &\sim \text{Binomial}(1, \pi) \\
S_{m,n} &\sim \text{Binomial}(1, \tau) \\
W_{m,n} &\sim \text{Multinomial}(L_{m,n}, \theta)
\end{aligned}$$

The parameters π and θ have the same semantics as in the naïve Bayes model. S is naturally modeled as a binomial variable, where τ is the parameter of S. S represents how likely it is that a sentence strongly conveys a perspective. We call this model the Latent Sentence Perspective Model (LSPM) because S is not directly observed. The graphical model representation of LSPM is shown in Figure 2.

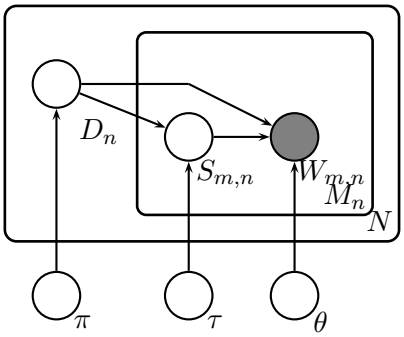


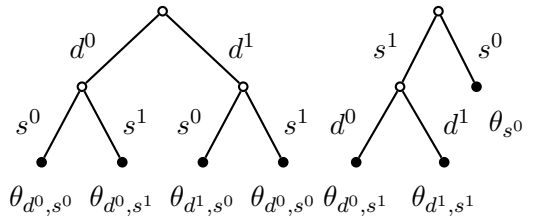
Figure 2: Latent Sentence Perspective Model

To use LSPM to identify the perspective of a new document $\tilde{D}$ with unknown sentence perspectives $\tilde{S}$, we calculate posterior probabilities by summing out possible combinations of sentence perspective in the document and parameters.

$$\int\int\int \sum_{S_{m,n}} \sum_{\tilde{S}} P(\tilde{D}, S_{m,n}, \tilde{S}, \pi, \tau, \theta | \{(D_n, W_n)\}_{n=1}^{N}, \tilde{W}) d\pi d\tau d\theta \quad (8)$$

As before, we resort to MCMC methods to sample from the posterior distributions, given in Equations (5) and (6).

As is often encountered in mixture models, there is an identifiability issue in LSPM. Because the values of S can be permuted without changing the likelihood function, the meanings of s^0 and s^1 are ambiguous. In Figure 3a, four θ values are used to represent the four possible combinations of document perspective d and sentence perspective intensity s. If we do not impose any constraints, s^1 and s^0 are exchangeable, and we can no longer strictly interpret s^1 as indicating a strong sentence-level perspective and s^0 as indicating that a sentence carries little or no perspective information. The other problem of this parameterization is that any improvement from LSPM over the naïve Bayes model is not necessarily



(a) s^0 and s^1 are not identifiable (b) sharing θ_{d^1,s^0} and θ_{d^0,s^0}

Figure 3: Two different parameterization of θ

due to the explicit modeling of sentence-level perspective. S may capture aspects of the document collection that we never intended to model. For example, s^0 may capture the editors' writing styles and s^1 the guests' writing styles in the `bitterlemons` corpus.

We solve the identifiability problem by forcing θ_{d^1,s^0} and θ_{d^0,s^0} to be identical and reducing the number of θ parameters to three. As shown in Figure 3b, there are separate θ parameters conditioned on the document perspective (left branch of the tree, d^0 is Israeli and d^1 is Palestinian), but there is single θ parameter when $S = s^0$ shared by both document-level perspectives (right branch of the tree). We assume that the sentences with little or no perspective information, i.e., $S = s^0$, are generated independently of the perspective of a document. In other words, sentences that are presenting common background information or introducing an issue and that do not strongly convey any perspective should look similar whether they are in Palestinian or Israeli documents. By forcing this constraint, we become more confident that s^0 represents sentences of little perspectives and s^1 represents sentences of strong perspectives from d^1 and d^0 documents.

5 Experiments

5.1 Identifying Perspective at the Document Level

We evaluate three different models for the task of identifying perspective at the document level: two naïve Bayes models (NB) with different inference methods and Support Vector Machines (SVM) (Cristianini and Shawe-Taylor, 2000). NB-B uses full Bayesian inference and NB-M uses Maximum a posteriori (MAP). We compare NB with SVM not only because SVM has been very effective for classifying topical documents (Joachims, 1998), but also to contrast generative models like NB with discriminative models like SVM. For training SVM, we represent each document as a V-dimensional feature vector, where V is the vocabulary size and each coordinate is the normalized term frequency within the document. We use a linear kernel for SVM and search for the best parameters using grid methods.

To evaluate the statistical models, we train them on the documents in the `bitterlemons` corpus and calculate how accurately each model predicts document perspective in ten-fold cross-validation experiments. Table 2 reports the average classification accuracy across the the 10 folds for each model. The accuracy of a baseline classifier, which randomly assigns the perspective of a document as Palestinian or Israeli, is 0.5, because there are equivalent numbers of documents from the two perspectives.

Model	Data Set	Accuracy	Reduction
Baseline		0.5	
SVM	Editors	0.9724	
NB-M	Editors	0.9895	61%
NB-B	Editors	0.9909	67%
SVM	Guests	0.8621	
NB-M	Guests	0.8789	12%
NB-B	Guests	0.8859	17%

Table 2: Results for Identifying Perspectives at the Document Level

The last column of Table 2 is error reduction relative to SVM. The results show that the naïve Bayes models and SVM perform surprisingly well on both the Editors and Guests subsets of the `bitterlemons` corpus. The naïve Bayes models perform slightly better than SVM, possibly because generative models (i.e., naïve Bayes models) achieve optimal performance with a smaller number of training examples than discriminative models (i.e., SVM) (Ng and Jordan, 2002), and the size of the `bitterlemons` corpus is indeed small. NB-B, which performs full Bayesian inference, improves

on NB-M, which only performs point estimation. The results suggest that the choice of words made by the authors, either consciously or subconsciously, reflects much of their political perspectives. Statistical models can capture word usage well and can identify the perspective of documents with high accuracy.

Given the performance gap between Editors and Guests, one may argue that there exist distinct editing artifacts or writing styles of the editors and guests, and that the statistical models are capturing these things rather than "perspectives." To test if the statistical models truly are learning perspectives, we conduct experiments in which the training and testing data are mismatched, i.e., from different subsets of the corpus. If what the SVM and naïve Bayes models learn are writing styles or editing artifacts, the classification performance under the mismatched conditions will be considerably degraded.

Model	Training	Testing	Accuracy	
Baseline			0.5	
SVM	Guests	Editors	0.8822	
NB-M	Guests	Editors	0.9327	43%
NB-B	Guests	Editors	0.9346	44%
SVM	Editors	Guests	0.8148	
NB-M	Editors	Guests	0.8485	18%
NB-B	Editors	Guests	0.8585	24%

Table 3: Identifying Document-Level Perspectives with Different Training and Testing Sets

The results on the mismatched training and testing experiments are shown in Table 3. Both SVM and the two variants of naïve Bayes perform well on the different combinations of training and testing data. As in Table 2, the naïve Bayes models perform better than SVM with larger error reductions, and NB-B slightly outperforms NB-M. The high accuracy on the mismatched experiments suggests that statistical models are not learning writing styles or editing artifacts. This reaffirms that document perspective is reflected in the words that are chosen by the writers.

We list the most frequent words (excluding stopwords) learned by the the NB-M model in Table 4. The frequent words overlap greatly between the Palestinian and Israeli perspectives, including "state," "peace," "process," "secure" ("security"), and "govern" ("government"). This is in contrast to what we expect from topical text classification (e.g., "Sports" vs. "Politics"), in which frequent words seldom overlap. Authors from different perspectives often choose words from a similar vocabulary but emphasize them differently. For example, in documents that are written from the Palestinian perspective, the word "palestinian" is mentioned more frequently than the word "israel." It is, however, the reverse for documents that are written from the Israeli perspective. Perspectives are also expressed in how frequently certain people ("sharon" v.s. "arafat"), countries ("international" v.s. "america"), and actions ("occupation" v.s. "settle") are mentioned. While one might solicit these contrasting word pairs from domain experts, our results show that statistical models such as SVM and naïve Bayes can automatically acquire them.

5.2 Identifying Perspectives at the Sentence Level

In addition to identifying the perspective of a document, we are interested in knowing which sentences of the document strongly conveys perspective information. Sentence-level perspective annotations do not exist in the `bitterlemons` corpus, which makes estimating parameters for the proposed Latent Sentence Perspective Model (LSPM) difficult. The posterior probability that a sentence strongly covey a perspective (Example (6)) is of the most interest, but we can not directly evaluate this model without gold standard annotations. As an alternative, we evaluate how accurately LSPM predicts the perspective of a document, again using 10-fold cross validation. Although LSPM predicts the perspective of both documents and sentences, we will doubt the quality of the sentence-level predictions if the document-level predictions are incorrect.

The experimental results are shown in Table 5. We include the results for the naïve Bayes models from Table 3 for easy comparison. The accuracy of LSPM is comparable or even slightly better than that of the naïve Bayes models. This is very encouraging and suggests that the proposed LSPM closely captures how perspectives are reflected at both the document and sentence levels. Examples 1 and 2 from the introduction were predicted by LSPM as likely to

Palestinian	palestinian, israel, state, politics, peace, international, people, settle, occupation, sharon, right, govern, two, secure, end, conflict, process, side, negotiate
Israeli	israel, palestinian, state, settle, sharon, peace, arafat, arab, politics, two, process, secure, conflict, lead, america, agree, right, gaza, govern

Table 4: The top twenty most frequent stems learned by the NB-M model, sorted by $P(w|d)$

Model	Training	Testing	Accuracy
Baseline			0.5
NB-M	Guests	Editors	0.9327
NB-B	Guests	Editors	0.9346
LSPM	Guests	Editors	0.9493
NB-M	Editors	Guests	0.8485
NB-B	Editors	Guests	0.8585
LSPM	Editors	Guests	0.8699

Table 5: Results for Perspective Identification at the Document and Sentence Levels

contain strong perspectives, i.e., large $\Pr(\tilde{S} = s^1)$. Examples 3 and 4 from the introduction were predicted by LSPM as likely to contain little or no perspective information, i.e., high $\Pr(\tilde{S} = s^0)$.

The comparable performance between the naïve Bayes models and LSPM is in fact surprising. We can train a naïve Bayes model directly on the sentences and attempt to classify a sentence as reflecting either a Palestinian or Israeli perspective. A sentence is correctly classified if the predicted perspective for the sentence is the same as the perspective of the document from which it was extracted. Using this model, we obtain a classification accuracy of only 0.7529, which is much lower than the accuracy previously achieved at the document level. Identifying perspectives at the sentence level is thus more difficult than identifying perspectives at the document level. The high accuracy at the document level shows that LSPM is very effective in pooling evidence from sentences that individually contain little perspective information.

6 Conclusions

In this paper we study a new problem of learning to identify the perspective from which a text is written at the document and sentence levels. We show that much of a document's perspective is expressed in word usage, and statistical learning algorithms such as SVM and naïve Bayes models can successfully uncover the word patterns that reflect author perspective with high accuracy. In addition, we develop a novel statistical model to estimate how strongly a sentence conveys perspective, in the absence of sentence-level annotations. By introducing latent variables and sharing parameters, the Latent Sentence Perspective Model is shown to capture well how perspectives are reflected at the document and sentence levels. The small but positive improvement due to sentence-level modeling in LSPM is encouraging. In the future, we plan to investigate how consistently LSPM sentence-level predictions are with human annotations.

Acknowledgment

This material is based on work supported by the Advanced Research and Development Activity (ARDA) under contract number NBCHC040037.

A Gibbs Samplers

Based the model specification described in Section 4.2 we derive the Gibbs samplers (Chen et al., 2000) for the Latent Sentence Perspective Model as follows,

$$\pi^{(t+1)} \sim \text{Beta}(\alpha_\pi + \sum_{n=1}^{N} d_n + \tilde{d}^{(t+1)}, \beta_\pi + N - \sum_{n=1}^{N} d_n + 1 - \tilde{d}^{(t+1)})$$

$$\tau^{(t+1)} \sim \text{Beta}(\alpha_\tau + \sum_{n=1}^{N}\sum_{m=1}^{M_n} s_{m,n} + \sum_{m=1}^{\tilde{M}} \tilde{s_m}, \beta_\tau + \sum_{n=1}^{N} M_n - \sum_{n=1}^{N}\sum_{m=1}^{M_n} s_{m,n} + \tilde{M} - \sum_{m=1}^{\tilde{M}} \tilde{s_m})$$

$$\theta^{(t+1)} \sim \text{Dirichlet}(\alpha_\theta + \sum_{n=1}^{N}\sum_{m=1}^{M_n} w_{m,n})$$

$$\Pr(S_{n,m}^{(t+1)} = s^1) \propto P(W_{m,n}|S_{m,n} = 1, \theta^{(t)}) \Pr(S_{m,n}^{(t+1)} = 1|\tau, D_n)$$

$$\Pr(\tilde{D}^{(t+1)} = d^1) \propto \prod_{m=1}^{\tilde{M}} dbinom(\tau_d^{(t+1)}) \prod_{m=1}^{\tilde{M}} dmultinom(\theta_{d,\tilde{m}^{(t)}}) dbinom(\pi^{(t)})$$

where dbinom and dmultinom are the density functions of binomial and multinomial distributions, respectively. The superscript t indicates that a sample is from the t-th iteration. We run three chains and collect 5000 samples. The first half of burn-in samples are discarded.

References

Philip Beineke, Trevor Hastie, and Shivakumar Vaithyanathan. 2004. The sentimental factor: Improving review classification via human-provided information. In *Proceedings of ACL-2004*.

Ming-Hui Chen, Qi-Man Shao, and Joseph G. Ibrahim. 2000. *Monte Carlo Methods in Bayesian Computation*. Springer-Verlag.

Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of WWW-2003*.

Michael L. Geis. 1987. *The Language of Politics*. Springer.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of KDD-2004*.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of ECML-1998*.

David D. Lewis. 1998. Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-1998*.

S. Morinaga, K. Yamanishi, K. Tateishi, and T. Fukushima. 2002. Mining product reputations on the web. In *Proceedings of KDD-2002*.

Tony Mullen and Nigel Collier. 2004. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of EMNLP-2004*.

T. Nasukawa and J. Yi. 2003. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of K-CAP 2003*.

Andrew Y. Ng and Michael Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS-2002*, volume 15.

Zhongdang Pan, Chin-Chuan Lee, Joseph Man Chen, and Clement Y.K. So. 1999. One event, three stories: Media narratives of the handover of hong kong in cultural china. *Gazette*, 61(2):99–112.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of EMNLP-2002*.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP-2005*, pages 339–346.

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of EMNLP-2003*.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of CoNLL-2003*.

Peter Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM TOIS*, 21(4):315–346.

T.A. van Dijk. 1988. *News as Discourse*. Lawrence Erlbaum, Hillsdale, NJ.

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computational Linguistics*, 30(3).

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP-2005*.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of EMNLP-2003*.

Unsupervised Grammar Induction by Distribution and Attachment

David J. Brooks
School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK
d.j.brooks@cs.bham.ac.uk

Abstract

Distributional approaches to grammar induction are typically inefficient, enumerating large numbers of candidate constituents. In this paper, we describe a simplified model of distributional analysis which uses heuristics to reduce the number of candidate constituents under consideration. We apply this model to a large corpus of over 400000 words of written English, and evaluate the results using EVALB. We show that the performance of this approach is limited, providing a detailed analysis of learned structure and a comparison with actual constituent-context distributions. This motivates a more structured approach, using a process of attachment to form constituents from their distributional components. Our findings suggest that distributional methods do not generalize enough to learn syntax effectively from raw text, but that attachment methods are more successful.

1 Introduction

Distributional approaches to grammar induction exploit the principle of substitutability: constituents of the same type may be exchanged with one another without affecting the syntax of the surrounding context. Reversing this notion, if we can identify "surrounding context" by observation, we can hypothesize that word sequences occurring in that context will be constituents of the same type. Thus, distributional methods can be used to segment text into constituents and classify the results. This work focuses on distributional learning from raw text.

Various models of distributional analysis have been used to induce syntactic structure, but most use probabilistic metrics to decide between candidate constituents. We show that the efficiency of these systems can be improved by exploiting some properties of probable constituents, but also that this reliance on probability is problematic for learning from text. As a consequence, we propose an extension to strict distributional learning that incorporates more information about constituent boundaries.

The remainder of this paper describes our experiences with a heuristic system for grammar induction. We begin with a discussion of previous distributional approaches to grammar induction in Section 2 and describe their implications in Section 3. We then introduce a heuristic distributional system in Section 4, which we analyze empirically against a treebank. Poor system performance leads us to examine actual constituent-context distributions (Section 5), the implications of which motivate a more structured extension to our learning system, which we describe and analyze in Section 6.

2 Previous approaches

Distributional methods analyze text by *alignment*, aiming to find equivalence classes covering substitutable units. We align common portions of texts termed *contexts*, leaving distinct contiguous word-sequences, termed *expressions*. An expression and its context form an *alignment pattern*, which is de-

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 117–124, New York City, June 2006.

fined as:

$$C_{left} \mid \textit{Expression} \mid C_{right} \quad \text{(AP1)}$$

From this alignment pattern, we can extract context-free grammar rules:

$$NT \rightarrow Expression_1 \vee ... \vee Expression_n \quad (1)$$

While the definition of expression is straightforward, the definition of context is problematic. We would like as much context as possible, but word-sequence contexts become less probable as their length increases, making learning harder. Therefore, simple models of context are preferred, although the precise definition varies between systems.

Distributional approaches to grammar induction fall into two categories, depending on their treatment of nested structure. The first category covers Expectation-Maximization (EM) systems. These systems propose constituents based on analysis of text, then select a *non-contradictory combination* of constituents for each sentence that maximizes a given metric, usually parsing probability. EM has the advantage that constituent probabilities are only compared when constituents compete, which removes the inherent bias towards shorter constituents, which tend to have higher probability. However, EM methods are more susceptible to data sparsity issues associated with raw text, because there is no generalization during constituent proposal.

Examples of EM learning systems are Context Distribution Clustering (CDC) (Clark, 2001) and Constituent-Context Model (CCM) (Klein, 2005, Chapter 5), which avoid the aforementioned data-sparsity issues by using a part-of-speech (POS) tagged corpus, rather than raw text. Alignment Based Learning (ABL) (van Zaanen, 2000) is the only EM system applied directly to raw text. ABL uses minimal String-Edit Distance between sentences to propose constituents, from which the most probable combination is chosen. However, ABL is relatively inefficient and has only been applied to small corpora.

The second category is that of incremental learning systems. An incremental system analyzes a corpus in a bottom-up fashion: each time a new constituent type is found, it is inserted into the corpus to provide data for later learning. This has the advantage of easing the data-sparsity issues described above because infrequent sequences are clustered into more frequent non-terminal symbols. However, in incremental systems, constituents are compared directly, which can lead to a bias towards shorter constituents.

The EMILE system (Adriaans, 1999) learns *shallow* languages in an incremental manner, and has been applied to natural language under the assumption that such languages are shallow. Shallowness is the property whereby, for any constituent type in a language, there exist well-supported minimal units of that type. EMILE aligns complete sentences only, attempting to isolate minimal units, which are then used to process longer sequences. This method is efficient because alignment is non-recursive. However, as a consequence, EMILE offers only a limited treatment of nested and recursive structures.

A more comprehensive approach to learning nested structure is found in the ADIOS system (Solan et al., 2003). ADIOS enumerates all patterns of a given length, under the condition that each sequence must have non-empty contexts and expressions. These patterns are ranked using an information gain metric, and the best pattern at each iteration is rewritten into the graph, before pattern scanning begins again. ADIOS learns context-sensitive equivalence classes, but does not induce grammars, and has not been formally evaluated against treebanks.

Grammar induction systems are evaluated using standard metrics for parser evaluation, and in particular, the EVALB algorithm[1]. The above systems have been evaluated with respect to the ATIS treebank. Compared with supervised parsers, these systems perform relatively poorly, with the strictly unsupervised EMILE and ABL systems recovering 16.8% and 35.6% of constituent structure respectively. The partially-supervised systems of CDC and CCM perform better, with the latter retrieving 47.6% of the constituent structure in ATIS. However, the strictly unsupervised systems of ABL, EMILE and ADIOS have not been evaluated on larger corpora, in part due to efficiency constraints.

[1]There are known issues with parser evaluation, although a discussion of these issues is outside the scope of this paper, and the reader is referred to (Klein, 2005, Chapter 2). We assume the standard evaluation for comparison with previous work.

3 Issues for distributional learning

There are many issues with distributional learning, especially when learning from raw text. First, previous systems hypothesize and select constituents according to the probability of their contexts: ABL, EMILE and CCM use the probability of proposed equivalence classes, or the equivalent context probability; ADIOS uses an information gain metric, again favouring probable contexts. However, when learning from raw text, this preference for hypotheses with more probable contexts means that open-class words will seldom be considered as contexts. In POS-based learners, it is possible to align open-class POS contexts. These contexts are demonstrably important despite low word probabilities, which suggests that selecting contexts on the basis of probability will be limited in success.

The second problem relates to word-senses. Alignment proceeds by matching orthographic types, but these types can have numerous associated syntactic senses. For example, 'to' plays two distinct roles: infinitive marker or preposition. If we align using the orthographic type, we will often misalign words, as seen in the following alignment:

I gave it	to	the man	in	the grey jacket
John agreed	to	see me	in	20 minutes

Here, we are (mis)aligning a prepositional 'to', with an infinitive marker. The result would be a correctly identified noun-phrase, 'the man', and an incorrect structure, contradicting both the verb-group 'to see' and the noun-phrase 'me'. This problem does not affect POS-based learning systems, as POS tags are unambiguously assigned.

Finally, grammar induction systems are typically inefficient, which prohibits training over large corpora. Distributional analysis is an expensive procedure, and must be performed for large numbers of word sequences. Previous approaches have tended to enumerate all alignment patterns, of which the best are selected using probabilistic metrics. However, given the preference for probable alignments, there is considerable wasted computation here, and it is on this issue that we shall focus.

4 A heuristic approach to alignment

Rather than enumerating all word sequences in a corpus, we propose a heuristic for guiding distributional systems towards more favourable alignment patterns, in a system called *Directed Alignment*. In this system, we define context as the ordered pair of left- and right-context for a given constituent, $\langle C_{left} - C_{right} \rangle$, where C_{left} and C_{right} are single-units. The atomic units of this system are words, but learned constituents may also act as context-units.

The probability of a pattern depends primarily on its contexts, since they are common to all matching sequences. We can reduce the task of finding probable alignments to simply finding probable context-pairs. However, we can reduce this further: for a context-pair to be probable, its components must also be probable. Therefore, rather than enumerating all patterns in the corpus, we direct the alignment procedure towards patterns where C_{left} and C_{right} are probable.

The first stage of direction creates an index for the corpus, compiling a list of unit types, where units are initially words. From this list of types, the most probable 1% are selected as *context-units*. These context-units are the only types allowed to fill the roles C_{left} and C_{right} in alignment patterns.

Alignments are created directly from the context-unit index. For each context-unit token cu in the index, we locate cu in the corpus and create an alignment pattern, such that cu is the left context (C_{left}). Next, we scan the sequence of words following cu, extending the alignment pattern until another context-unit cu' is found, or a fixed length threshold is exceeded. If cu' is found, it fills the role of right context (C_{right}), and the completed alignment pattern is cached; otherwise, the pattern is disregarded.

Direction permits two forms of valid expressions in the context $\langle cu - cu' \rangle$:

1. $nc_1 \ldots nc_n$, where each nc_i is a non-context
2. $c_1 \ldots c_n$, where each c_i is a context-unit

The first of these forms allows us to examine non-nested alignments. The second allows us to analyze nested alignments only after inner constituents have been learned. These constraints reduce the number of constituents under consideration at any time to a manageable level. As a result, we can scan very large numbers of alignment patterns with relatively little overhead.

As an example, consider the following sequence, with context units underlined:

> put the whole egg , all the seasonings and vegetables into the bowl and process for 10 seconds until smoothly pured .

This would be broken into non-recursive expressions[2]:

> (put) the (whole egg) , all the (seasonings) and (vegetables) into the (bowl) and (process) for (10 seconds) until (smoothly pureed) .

These expressions will be replaced by non-terminal unit representing the class of expressions, such that each class contains all units across the corpus that occur in the same context:

> NT0 the NT1 , all the NT2 and NT3 into the NT2 and NT4 for NT5 until NT6 .

Following this generalization nested structures can be discovered using the same process.

This approach has some interesting parallels with chunking techniques, most notably that of function-word phrase identification (Smith and Witten, 1993). This similarity is enforced by disallowing nested structures. Unlike chunking systems, however, this work will also attempt to recover nested structures by means of incremental learning.

4.1 Selecting alignment patterns

The direction process extracts a set of candidate alignments, and from this set we select the best alignment to rewrite as an equivalence class. Previous approaches offer a number of metrics for ranking constituents, based around constituent or context probability (ABL and CCM), Mutual Information (CDC), and information gain (ADIOS). We have implemented several of these metrics, but our experiences suggest that context probability is the most successful.

The probability of an alignment is effectively the sum of all path probabilities through the alignment:

$$P(C_{left}, C_{right}) = \Sigma P(path_{left,right}) \quad (2)$$

where each $path_{left,right}$ is a unique word sequence starting with $left$ and ending with $right$, under the constraints on expressions described above. There is an important practical issue here: probability sums such as that in Equation 2 do not decrease when expressions are replaced with equivalence classes. To alleviate this problem, we rewrite the units when updating the distribution, but discard paths that match the current alignment. This prevents looping while allowing the rewritten paths to contribute to nested structures.

[2]For clarity, we have shown all alignments for the given sentence simultaneously. However, the learning process is incremental, so each alignment would be proposed during a distinct learning iteration.

4.2 Generalizing expression classes

The model outlined above is capable of learning strictly context-sensitive constituents. While this does allow for nested constituents, it is problematic for generalization. Consider the following equivalence classes, which are proposed relatively early in Directed Alignment:

the	NT1	of
the	NT2	in

Here, the non-terminals have been assigned on the basis of context-pairs: NT1 is defined by $\langle the - of \rangle$ and NT2 is defined by $\langle the - in \rangle$. These types are distinct, although intuitively they account for simple noun-phrases. If we then propose an alignment pattern with NT1 as C_{left}, it must be followed by 'of', which removes any possibility of generalizing 'of' and 'in'.

We alleviate this problem by generalizing equivalence classes, using a simple clustering algorithm. For each new alignment, we compare the set of expressions with all existing expression classes, ranking the comparisons by the degree of overlap with the current alignment. If this degree of overlap exceeds a fixed threshold, the type of the existing class is assumed; otherwise, a new class is created.

4.3 Experiments, results and analysis

To evaluate our algorithm, we follow the standard approach of comparing the output of our system with that of a treebank. We use the EVALB algorithm, originally designed for evaluating supervised parsing systems, with identical configuration to that of (van Zaanen, 2000). However, we apply our algorithms to a different corpus: the written sub-corpus of the International Corpus of English, Great Britain Component (henceforth ICE-GB), with punctuation removed. This consists of 438342 words, in 22815 sentences. We also include a baseline instantiation

System	UP	UR	F_1	CB
FWB	30.0	11.0	16.0	0.36
DA	23.3	8.0	11.9	0.30
$DA_{cluster}$	23.6	8.1	12.0	0.30

Table 1: EVALB results after 500 iterations of Directed Alignment applied to ICE-GB, showing both context-sensitive (DA) and clustered ($DA_{cluster}$) alignment. The columns represent Unlabeled Precision, Unlabeled Recall, Unlabeled F-Score and the proportion of sentence with crossing brackets respectively.

of our algorithm, which chunks text into expressions between function words, which we refer to as Function-Word Bracketing (FWB).

Table 1 summarizes the EVALB scores for two 500-iteration runs of Directed Alignment over ICE-GB: DA is the standard context-sensitive version of the algorithm; $DA_{cluster}$ is the version with context clustering. FWB precision is relatively low, with only 30% of proposed structures appearing in the treebank. Recall is even lower, with only 11% of structure retrieved. This is unsurprising, as no nested constructions are considered.

In comparison, both versions of Directed Alignment perform significantly worse, with $DA_{cluster}$ being only fractionally better than standard DA. Experiments over more learning iterations suggest that the performance of DA converges on FWB, with few nested constituents discovered. Both variants of the system produce very poor performance, with very little nested structure recovered. While these results seem discouraging, it is worth investigating system performance further.

Table 2, summarizes the success of the algorithm at discovering different types of constituent. Note that these results are unlabeled, so we are examining the proportion of each type of constituent in ICE-GB that has been identified. Here, Directed Alignment exhibits the most success at identifying non-clauses, of which the primary source of success is short sentence fragments. Around 10% of noun-phrases (NP), verb-phrases (VP) and subordinate-phrases (SUBP) were recovered, this limited success reflects the nature of the constituents: all three have relatively simple constructions, whereby a single word represents the constituent. In contrast, constituent types that comprise multiple units, such as prepositional-phrases (PP), are seldom recovered.

Category	Frequency	Recall (%)		
		FWB	DA	$DA_{cluster}$
NP	117776	11.81	10.83	10.79
CL	28641	0.50	1.21	1.14
VP	50280	20.88	9.58	9.89
PP	42134	0.10	0.67	0.73
SUBP	7474	1.10	11.05	11.15
NONCL	1919	4.27	22.98	22.98

Table 2: Constituent retrieval results for Function-Word Bracketing (FWB) and Directed Alignment (DA and $DA_{cluster}$), categorized by gold-type

(a) DA, top 5 noun-matches of 271

Learned	Recall	Precision
NT0	4.61	84.53
NT5	1.58	93.44
NT7	1.36	87.14
NT4	1.09	75.10
NT10	0.82	84.54

(b) $DA_{cluster}$, top 5 noun-matches of 135

Learned	Recall	Precision
NT0	6.93	87.09
NT4	6.48	89.91
NT8	2.62	40.48
NT11	0.86	68.60
NT10	0.58	16.95

Table 3: The top five expression classes to match N (noun) in ICE-GB, ranked by recall.

4.3.1 Class generalization

During learning in $DA_{cluster}$, we induce generalized classes using the expression clustering algorithm. This generalization can be evaluated, comparing induced classes with those in the treebank using precision and recall. Table 2(a) shows the top five proposed classes matching the type noun (N) in ICE-GB during 500 iterations of context-sensitive Directed Alignment. There are 271 types matching noun, and as can be seen, the top five account for a very small proportion of all nouns, some 9.46% (recall).

Table 2(b) shows the same analysis for Directed Alignment with class generalization. For noun matches, we can see that there are far fewer proposed classes (135), and that those classes are much more probable, the top five accounting for 17.47%

(a) Noun Phrases (frequency=123870)

LEFT		START		END		RIGHT	
SYMB	REC	SYMB	REC	SYMB	REC	SYMB	REC
PREP	0.36	ART	0.29	N	0.53	PUNC	0.36
V	0.19	PRON	0.29	PRON	0.19	V	0.18
#STA#	0.12	N	0.2	N_2	0.11	AUX	0.13
CONJ	0.11	N_1	0.06	PUNC	0.06	CONJ	0.09
PUNC	0.09	ADJ	0.06	NUM	0.04	PREP	0.07

(b) Verb Phrases (frequency=50693)

Left		Start		End		Right	
SYMB	REC	SYMB	REC	SYMB	REC	SYMB	REC
PRON	0.32	V	0.68	V	0.98	PREP	0.20
N	0.26	AUX	0.29	PUNC	0.01	ART	0.16
PTCL	0.11	AUX_1	0.02	AUX	0.00	PRON	0.14
PUNC	0.06	V_1	0.00	V_2	0.00	ADV	0.13
CONJ	0.05	ADV	0.00	ADV	0.00	ADJ	0.09

(c) Prepositional Phrases (frequency=45777)

Left		Start		End		Right	
SYMB	REC	SYMB	REC	SYMB	REC	SYMB	REC
N	0.46	PREP	0.96	N	0.63	PUNC	0.56
V	0.23	PREP_1	0.02	N_2	0.12	CONJ	0.09
ADV	0.05	ADV	0.01	PUNC	0.08	PREP	0.09
PUNC	0.05	NUM	0.00	PRON	0.05	V	0.07
ADJ	0.04	ADV_1	0.00	NUM	0.03	AUX	0.05

Table 4: The five most frequent left/start/end/right POS contexts for NP, VP and PP constituents.

of nouns in ICE-GB. The algorithm seems to be achieving some worthwhile generalization, which is reflected in a slight increase in EVALB scores for $DA_{cluster}$. However, this increase is not a significant one, suggesting that this generalization is not sufficient to support distributional learning. We might expect this: attempting to cluster based on the low-frequency and polysemous words in expressions seems likely to produce unreliable clusters.

5 A closer look at distributional contexts

The results discussed so far seem discouraging for the approach. However, there are good reasons why these results are so poor, and why we can expect little improvement in the current formulation. We can show some of these reasons by examining actual constituent-context distributions.

Table 4 shows an analysis of the constituent types NP, VP and PP in ICE-GB, against the five most frequent POS tags[3] occurring as left-context, constituent-start, constituent-end, and right-context. We distinguish the following POS categories as being primarily functional, as they account for the majority of context-units considered by Directed Alignment: prepositions (PREP), articles (ART), auxiliaries (AUX), sentence-starts (#STA#), pronouns (PRON), conjunctions (CONJ), particles (PTCL) and punctuation (PUNC).

From Table 4, we can see that noun-phrases and verb-phrases are relatively well-suited to our approach. First, both types have strong functional left- and right-contexts: 58% of NP left-contexts and 50% of NP right-contexts are members of our functional POS; similarly, 43% of VP left-contexts and 49% of VP right-contexts are functional. This means that a probability-based model of context, such as ours, will find relatively strong support for these types. Second, both NP and VP have minimal unit types: nouns and pronouns for NP; verbs for VP. As a consequence, these types tend to carry more probability mass, since shorter sequences tend to be more frequent. We should expect our system to perform reasonably on NP and VP as a result.

In contrast, prepositional-phrases are much less amenable to distributional analysis. First, PP tend to be longer, since they contain NP, and this has obvious repercussions for alignment probabilities. More damagingly, PP contexts are dominated by open-class words - the top 74% of PP left-contexts are nouns, verbs and adverbs. Therefore, a purely probabilistic distributional approach cannot account for prepositional-phrases, since learning data is too sparse. Previous approaches have relied upon open-class generalization to reduce this problem, but these methods suffer from the same problems of data sparsity, and as such are not reliable enough to resolve the issue.

6 Attachment

We have seen that strictly probabilistic distributional analysis is not sufficient to learn constituents from raw text. If we are to improve upon this, we must find a way to identify constituents from their component parts, as well as by contextual analysis. The constituent-context distributions in Table 4 give us some clues as to where to start: both noun-phrases and prepositional-phrases show very significant constituent-starts, with articles and pronouns starting 58% of NP, and prepositions starting 94% of all PP. These functional types would be identified as contexts in Directed Alignment, but the strong relation to their containing constituents would be ig-

[3]The same trends can be shown for words, but a POS analysis is preferred for clarity and brevity.

nored.

One method for achieving such an internal relationship might be to attach contexts to the expressions with which they co-occur, and we propose using such a method here. However, this requires that we have some criterion for deciding when and how expressions should be attached to their contexts. We use a measure based on STOP arguments (Collins, 1999), which allows us to condition the decision to insert a constituent boundary on the evidence we see for doing so. For raw text, the only boundaries that are explicitly marked are at the start and end of sentences, and it is this information we use to decide when to attach contexts to expressions[4]. In other words, if a context is likely to start a sentence, we assume it is also likely to start a constituent at other positions within a sentence.

In order to calculate the likelihood of a particular context word w occurring at the start or end of a sentence, we simply use the bigram probabilities between w and the special symbols START and END, which denote the start and end of a sentence respectively. From these probabilities, we calculate Mutual Information $MI(START, w)$ and $MI(w, END)$. We prefer MI because it describes the strength of the relation between w and these special symbols without bias towards more probable words. From these MI values, we calculate a *Directional Preference* (DP) for the context word:

$$dp(w) = MI(w, END) - MI(START, w) \quad (3)$$

This yields a number representing whether w is more likely to start or end a sentence. This number will be zero if we are equally likely to see w at the start or end of a sentence, negative if w is more likely to start a sentence, and positive if w is is more likely to end a sentence.

Using DP, we can decide how to attach an expression to its contexts. For a given alignment, we consider the possibility of attaching the expression to neither context, the left-context, or the right-context, by comparing the DP for the left- and right-contexts. If the left-context shows a strong tendency to start sentences, and the right-context does not show a strong tendency to end sentences (i.e. there is an overall DP is negative), we attach the expression to its left-context; if the reverse situation is true, we attach the expression to its right context. Should the difference between these DP fall below a threshold, neither context is preferred, and the expression remains unattached.

Let us consider a specific example of attachment. The first alignment considered by the system (when applied to ICE-GB) is:

the NT1 of

Here, we need to compare the likelihood of seeing a constituent start with 'the' with with the likelihood of seeing a constituent end with 'of'. Intuitively, 'the' occurs frequently at the start of a sentence, and never at the end. Consequently, it has a high negative DP. Meanwhile 'of' has a small negative DP. In combination, there is a high negative DP, so we attach the expression to the left-context, 'the'.

[4]For this method to work, we assume that our corpus is segmented into sentences. This is not the case for speech, but for learning from text it seems a reasonable assumption.

System	UP	UR	F_1	CB
DA_{STOP}	33.6	14.1	19.8	0.42

Table 5: EVALB results after 500 iterations of Directed Alignment with STOP attachment applied to ICE-GB (DA_{STOP}).

Category	Frequency	Recall (%)
NP	117776	18.11
VP	50280	9.78
PP	42134	18.19
CL	28641	2.97
SUBP	7474	12.82
NONCL	1919	22.62

Table 6: Constituent retrieval results for DA_{STOP}, categorized by gold-type

6.1 Experimental Analysis

We applied Directed Alignment with attachment based on STOP arguments (DA_{STOP}) to ICE-GB as before, running for 500 iterations. These results are shown in Table 5. The results are encouraging. Unlabeled precision increased by almost 50%, from 23.6% for $DA_{cluster}$ to 33.6%. Likewise, system recall increased dramatically, from 8.1% to 14.1%, up some 75%. Crossing-brackets increased slightly, but remained relatively low at 0.42.

Table 6 shows the breakdown of EVALB scores

for the major non-terminal types, as before. The improvement in EVALB scores is attributable to a marked increase in success at identifying prepositional-phrases, with a lesser increase in noun-phrase identification.

6.2 Discussion

The attachment procedure described above is more successful at discovering nested constituents than distributional methods. There are good reasons why this should be the case. First, attachment compresses the corpus, removing the bias towards shorter sequences. Indeed, the algorithm seems capable of retrieving complex constituents of up to ten words in length during the first 500 iterations.

Second, the STOP-conditioning criterion, while somewhat *ad hoc* in relation to distributional methods, allows us to assess where constituent boundaries are likely to occur. As such, this can be seen as a rudimentary method for establishing argument relations, such as those observed in (Klein, 2005, Chapter 6).

Despite these improvements, the attachment process also makes some systematic mistakes. Some of these may be attributed to discrepancies between the syntactic theory used to annotate the treebank and the attachment process. For example, verbs are routinely attached to their subjects before objects, contradicting the more traditional interpretation present in treebanks. Some of the remaining mistakes can be attributed to the misalignment, due to the orthographic match problem described in Section 3.

7 Future Work

The major problem when applying distributional methods to raw text is that of orthographic matching, which causes misalignments between alternative senses of a particular word-form. To reduce this problem, context-units must be classified in some way to disambiguate these different senses. Such classification could be used as a precursor to alignment in the system we have described.

In addition, to better evaluate the quality of attachment, dependency representations and treebanks could be used, which do not have an explicit order on attachment. This would give a more accurate evaluation where subject-verb attachment is concerned.

8 Conclusions

We have presented an incremental grammar induction system that uses heuristics to improve the efficiency of distributional learning. However, in tests over a large corpus, we have shown that it is capable of learning only a small subset of constituent structure. We have analyzed actual constituent-context distributions to explain these limitations. This analysis provides the motivation for a more structured learning method, which incorporates knowledge of verifiable constituent boundaries - the starts and ends of sentences. This improved system performs significantly better, with a 75% increase in recall over distributional methods, and a significant improvement at retrieving structures that are problematic for distributional methods alone.

References

Pieter Adriaans. 1999. Learning shallow context-free languages under simple distributions. Technical Report PP-1999-13, Institute for Logic, Language, and Computation, Amsterdam.

Alexander Clark. 2001. Unsupervised induction of stochastic context free grammars with distributional clustering. In *Proceedings of the Fifth Conference on Natural Language Learning*, pages 105–112, Toulouse, France, July.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Dan Klein. 2005. *The Unsupervised Learning of Natural Language Structure*. Ph.D. thesis, Department of Computer Science, Stanford University, March.

Tony C. Smith and Ian H. Witten. 1993. Language inference from function words. Working Paper Series 1170-487X-1993/3, Department of Computer Science, University of Waikato, Hamilton, New Zealand, August.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. 2003. Unsupervised efficient learning and representation of language structures. In R. Alterman and D. Kirsch, editors, *Proceedings of the 25th Conference of the Cognitive Science Society*, Hillsdale, NJ. Erlbaum.

Menno van Zaanen. 2000. Learning structure using Alignment Based Learning. In *Proceedings of the Third Annual Doctoral Research Colloquium (CLUK)*, pages 75–82.

Learning Auxiliary Fronting with Grammatical Inference

Alexander Clark
Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX
alexc@cs.rhul.ac.uk

Rémi Eyraud
EURISE
23, rue du Docteur Paul Michelon
42023 Saint-Étienne Cedex 2
France
remi.eyraud@univ-st-etienne.fr

Abstract

We present a simple context-free grammatical inference algorithm, and prove that it is capable of learning an interesting subclass of context-free languages. We also demonstrate that an implementation of this algorithm is capable of learning auxiliary fronting in polar interrogatives (AFIPI) in English. This has been one of the most important test cases in language acquisition over the last few decades. We demonstrate that learning can proceed even in the complete absence of examples of particular constructions, and thus that debates about the frequency of occurrence of such constructions are irrelevant. We discuss the implications of this on the type of innate learning biases that must be hypothesized to explain first language acquisition.

1 Introduction

For some years, a particular set of examples has been used to provide support for nativist theories of first language acquisition (FLA). These examples, which hinge around auxiliary inversion in the formation of questions in English, have been considered to provide a strong argument in favour of the nativist claim: that FLA proceeds primarily through innately specified domain specific mechanisms or knowledge, rather than through the operation of general-purpose cognitive mechanisms. A key point of empirical debate is the frequency of occurrence of the forms in question. If these are vanishingly rare, or non-existent in the primary linguistic data, and yet children acquire the construction in question, then the hypothesis that they have innate knowledge would be supported. But this rests on the assumption that examples of that specific construction are necessary for learning to proceed. In this paper we show that this assumption is false: that this particular construction can be learned without the learner being exposed to any examples of that particular type. Our demonstration is primarily mathematical/computational: we present a simple experiment that demonstrates the applicability of this approach to this particular problem neatly, but the data we use is not intended to be a realistic representation of the primary linguistic data, nor is the particular algorithm we use suitable for large scale grammar induction.

We present a general purpose context-free grammatical algorithm that is provably correct under a certain learning criterion. This algorithm incorporates no domain specific knowledge: it has no specific information about language; no knowledge of X-bar schemas, no hidden sources of information to reveal the structure. It operates purely on unannotated strings of raw text. Obviously, as all learning algorithms do, it has an implicit learning bias. This very simple algorithm has a particularly clear bias, with a simple mathematical description, that allows a remarkably simple characterisation of the set of languages that it can learn. This algorithm does not use a statistical learning paradigm that has to be tested on large quantities of data. Rather it uses a

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 125–132, New York City, June 2006.

symbolic learning paradigm, that works efficiently with very small quantities of data, while being very sensitive to noise. We discuss this choice in some depth below.

For reasons that were first pointed out by Chomsky (Chomsky, 1975, pages 129–137), algorithms of this type are not capable of learning all of natural language. It turns out, however, that algorithms based on this approach are sufficiently strong to learn some key properties of language, such as the correct rule for forming polar questions.

In the next section we shall describe the dispute briefly; in the subsequent sections we will describe the algorithm we use, and the experiments we have performed.

2 The Dispute

We will present the dispute in traditional terms, though later we shall analyse some of the assumptions implicit in this description. In English, polar interrogatives (yes/no questions) are formed by fronting an auxiliary, and adding a dummy auxiliary "do" if the main verb is not an auxiliary. For example,

Example 1a The man is hungry.

Example 1b Is the man hungry?

When the subject NP has a relative clause that also contains an auxiliary, the auxiliary that is moved is not the auxiliary in the relative clause, but the one in the main (matrix) clause.

Example 2a The man who is eating is hungry.

Example 2b Is the man who is eating hungry?

An alternative rule would be to move the first occurring auxiliary, i.e. the one in the relative clause, which would produce the form

Example 2c Is the man who eating is hungry?

In some sense, there is no reason that children should favour the correct rule, rather than the incorrect one, since they are both of similar complexity and so on. Yet children do in fact, when provided with the appropriate context, produce sentences of the form of Example 2b, and rarely if ever produce errors of the form Example 2c (Crain and Nakayama, 1987). The problem is how to account for this phenomenon.

Chomsky claimed first, that sentences of the type in Example 2b are vanishingly rare in the linguistic environment that children are exposed to, yet when tested they unfailingly produce the correct form rather than the incorrect Example 2c. This is put forward as strong evidence in favour of innately specified language specific knowledge: we shall refer to this view as linguistic nativism.

In a special volume of the Linguistic Review, Pullum and Scholz (Pullum and Scholz, 2002), showed that in fact sentences of this type are not rare at all. Much discussion ensued on this *empirical* question and the consequences of this in the context of arguments for linguistic nativism. These debates revolved around both the methodology employed in the study, and also the consequences of such claims for nativist theories. It is fair to say that in spite of the strength of Pullum and Scholz's arguments, nativists remained completely unconvinced by the overall argument.

(Reali and Christiansen, 2004) present a possible solution to this problem. They claim that local statistics, effectively n-grams, can be sufficient to indicate to the learner which alternative should be preferred. However this argument has been carefully rebutted by (Kam et al., 2005), who show that this argument relies purely on a phonological coincidence in English. This is unsurprising since it is implausible that a flat, finite-state model should be powerful enough to model a phenomenon that is clearly structure dependent in this way.

In this paper we argue that the discussion about the rarity of sentences that exhibit this particular structure is irrelevant: we show that simple grammatical inference algorithms can learn this property even in the complete absence of sentences of this particular type. Thus the issue as to how frequently an infant child will see them is a moot point.

3 Algorithm

Context-free grammatical inference algorithms are explored in two different communities: in grammatical inference and in NLP. The task in NLP is normally taken to be one of recovering appropriate annotations (Smith and Eisner, 2005) that normally represent constituent structure (strong learning), while in grammatical inference, researchers

are more interested in merely identifying the language (weak learning). In both communities, the best performing algorithms that learn from raw positive data only [1], generally rely on some combination of three heuristics: frequency, information theoretic measures of constituency, and finally substitutability. [2] The first rests on the observation that strings of words generated by constituents are likely to occur more frequently than by chance. The second heuristic looks for information theoretic measures that may predict boundaries, such as drops in conditional entropy. The third method which is the foundation of the algorithm we use, is based on the distributional analysis of Harris (Harris, 1954). This principle has been appealed to by many researchers in the field of grammatical inference, but these appeals have normally been informal and heuristic (van Zaanen, 2000).

In its crudest form we can define it as follows: given two sentences "I saw a cat over there", and "I saw a dog over there" the learner will hypothesize that "cat" and "dog" are similar, since they appear in the same context "I saw a __ there". Pairs of sentences of this form can be taken as evidence that two words, or strings of words are substitutable.

3.1 Preliminaries

We briefly define some notation.

An *alphabet* Σ is a finite nonempty set of symbols called *letters*. A *string* w over Σ is a finite sequence $w = a_1 a_2 \dots a_n$ of letters. Let $|w|$ denote the length of w. In the following, letters will be indicated by $a, b, c, \dots$, strings by $u, v, \dots, z$, and the empty string by λ. Let Σ^* be the set of all strings, the free monoid generated by Σ. By a language we mean any subset $L \subseteq \Sigma^*$. The set of all substrings of a language L is denoted $Sub(L) = \{u \in \Sigma^+ : \exists l, r, lur \in L\}$ (notice that the empty word does not belong to $Sub(L)$). We shall assume an order $\prec$ or $\preceq$ on Σ which we shall extend to Σ^* in the normal way by saying that $u \prec v$ if $|u| < |v|$ or $|u| = |v|$ and u is lexicographically before v.

A grammar is a quadruple $G = \langle V, \Sigma, P, S \rangle$ where Σ is a finite alphabet of *terminal symbols*, V is a finite alphabet of *variables* or *non-terminals*, P is a finite set of *production rules*, and $S \in V$ is a start symbol.

If $P \subseteq V \times (\Sigma \cup V)^+$ then the grammar is said to be context-free (CF), and we will write the productions as $T \to w$.

We will write $uTv \Rightarrow uwv$ when $T \to w \in P$. $\stackrel{*}{\Rightarrow}$ is the reflexive and transitive closure of $\Rightarrow$.

In general, the definition of a class $\mathcal{L}$ relies on a class $\mathcal{R}$ of abstract machines, here called *representations*, together with a function $\mathcal{L}$ from representations to languages, that characterize all and only the languages of $\mathcal{L}$: (1) $\forall R \in \mathcal{R}, \mathcal{L}(R) \in \mathcal{L}$ and (2) $\forall L \in \mathcal{L}, \exists R \in \mathcal{R}$ such that $\mathcal{L}(R) = L$. Two representations R_1 and R_2 are *equivalent iff* $\mathcal{L}(R_1) = \mathcal{L}(R_2)$.

3.2 Learning

We now define our learning criterion. This is identification in the limit from positive text (Gold, 1967), with polynomial bounds on data and computation, but not on errors of prediction (de la Higuera, 1997).

A learning algorithm A for a class of representations $\mathcal{R}$, is an algorithm that computes a function from a finite sequence of strings $s_1, \dots, s_n$ to $\mathcal{R}$. We define a presentation of a language L to be an infinite sequence of elements of L such that every element of L occurs at least once. Given a presentation, we can consider the sequence of hypotheses that the algorithm produces, writing $R_n = A(s_1, \dots s_n)$ for the nth such hypothesis.

The algorithm A is said to identify the class $\mathcal{R}$ in the limit if for every $R \in \mathcal{R}$, for every presentation of $\mathcal{L}(R)$, there is an N such that for all $n > N$, $R_n = R_N$ and $\mathcal{L}(R) = \mathcal{L}(R_N)$.

We further require that the algorithm needs only polynomially bounded amounts of data and computation. We use the slightly weaker notion defined by de la Higuera (de la Higuera, 1997).

Definition A representation class $\mathcal{R}$ is identifiable in the limit from positive data with polynomial time and data iff there exist two polynomials $p(), q()$ and an algorithm A such that $S \subseteq \mathcal{L}(R)$

1. Given a positive sample S of size m A returns a representation $R \in \mathcal{R}$ in time $p(m)$, such that
2. For each representation R of size n there exists

[1] We do not consider in this paper the complex and contentious issues around negative data.

[2] For completeness we should include lexical dependencies or attraction.

a characteristic set CS of size less than $q(n)$ such that if $CS \subseteq S$, A returns a representation R' such that $\mathcal{L}(R) = \mathcal{L}(R')$.

3.3 Distributional learning

The key to the Harris approach for learning a language L, is to look at pairs of strings u and v and to see whether they occur in the same contexts; that is to say, to look for pairs of strings of the form lur and lvr that are both in L. This can be taken as evidence that there is a non-terminal symbol that generates both strings. In the informal descriptions of this that appear in Harris's work, there is an ambiguity between two ideas. The first is that they should appear in *all* the same contexts; and the second is that they should appear in *some* of the same contexts. We can write the first criterion as follows:

$$\forall l, r \; lur \in L \text{ if and only if } lvr \in L \qquad (1)$$

This has also been known in language theory by the name syntactic congruence, and can be written $u \equiv_L v$.

The second, weaker, criterion is

$$\exists l, r \; lur \in L \text{ and } lvr \in L \qquad (2)$$

We call this *weak substitutability* and write it as $u \doteq_L v$. Clearly $u \equiv_L v$ implies $u \doteq_L v$ when u is a substring of the language. Any two strings that do not occur as substrings of the language are obviously syntactically congruent but not weakly substitutable.

First of all, observe that syntactic congruence is a purely language theoretic notion that makes no reference to the grammatical representation of the language, but only to the set of strings that occur in it. However there is an obvious problem: syntactic congruence tells us something very useful about the language, but all we can observe is weak substitutability.

When working within a Gold-style identification in the limit (IIL) paradigm, we cannot rely on statistical properties of the input sample, since they will in general not be generated by random draws from a fixed distribution. This, as is well known, severely limits the class of languages that can be learned under this paradigm. However, the comparative simplicity of the IIL paradigm in the form when there are polynomial constraints on size of characteristic sets and computation(de la Higuera, 1997) makes it a suitable starting point for analysis.

Given these restrictions, one solution to this problem is simply to *define* a class of languages where substitutability implies congruence. We call these the substitutable languages: A language L is substitutable if and only if for every pair of strings u, v, $u \doteq_L v$ implies $u \equiv_L v$. This rather radical solution clearly rules out the syntax of natural languages, at least if we consider them as strings of raw words, rather than as strings of lexical or syntactic categories. Lexical ambiguity alone violates this requirement: consider the sentences "The rose died", "The cat died" and "The cat rose from its basket". A more serious problem is pairs of sentences like "John is hungry" and "John is running", where it is not ambiguity in the syntactic category of the word that causes the problem, but rather ambiguity in the context. Using this assumption, whether it is true or false, we can then construct a simple algorithm for grammatical inference, based purely on the idea that whenever we find a pair of strings that are weakly substitutable, we can generalise the hypothesized language so that they are syntactically congruent.

The algorithm proceeds by constructing a graph where every substring in the sample defines a node. An arc is drawn between two nodes if and only if the two nodes are weakly substitutable with respect to the sample, i.e. there is an arc between u and v if and only if we have observed in the sample strings of the form lur and lvr. Clearly all of the strings in the sample will form a clique in this graph (consider when l and r are both empty strings). The connected components of this graph can be computed in time polynomial in the total size of the sample. If the language is substitutable then each of these components will correspond to a congruence class of the language.

There are two ways of doing this: one way, which is perhaps the purest involves defining a reduction system or semi-Thue system which directly captures this generalisation process. The second way, which we present here, will be more familiar to computational linguists, and involves constructing a grammar.

3.4 Grammar construction

Simply knowing the syntactic congruence might not appear to be enough to learn a context-free grammar, but in fact it is. In fact given the syntactic congruence, and a sample of the language, we can simply write down a grammar in Chomsky normal form, and under quite weak assumptions this grammar will converge to a correct grammar for the language.

This construction relies on a simple property of the syntactic congruence, namely that is in fact a congruence: i.e.,

$$u \equiv_L v \text{ implies } \forall l, r \;\; lur \equiv_L lvr$$

We define the syntactic monoid to be the quotient of the monoid $\Sigma^* / \equiv_L$. The monoid operation $[u][v] = [uv]$ is well defined since if $u \equiv_L u'$ and $v \equiv_L v'$ then $uv \equiv_L u'v'$.

We can construct a grammar in the following trivial way, from a sample of strings where we are given the syntactic congruence.

- The non-terminals of the grammar are identified with the congruence classes of the language.
- For any string $w = uv$, we add a production $[w] \rightarrow [u][v]$.
- For all strings a of length one (i.e. letters of Σ), we add productions of the form $[a] \rightarrow a$.
- The start symbol is the congruence class which contains all the strings of the language.

This defines a grammar in CNF. At first sight, this construction might appear to be completely vacuous, and not to define any strings beyond those in the sample. The situation where it generalises is when two different strings are congruent: if $uv = w \equiv w' = u'v'$ then we will have two different rules $[w] \rightarrow [u][v]$ and $[w] \rightarrow [u'][v']$, since $[w]$ is the same non-terminal as $[w']$.

A striking feature of this algorithm is that it makes no attempt to identify which of these congruence classes correspond to non-terminals in the target grammar. Indeed that is to some extent an ill-posed question. There are many different ways of assigning constituent structure to sentences, and indeed some reputable theories of syntax, such as dependency grammars, dispense with the notion of constituent structure all together. De facto standards, such as the Penn treebank annotations are a somewhat arbitrary compromise among many different possible analyses. This algorithm instead relies on the syntactic monoid, which expresses the combinatorial structure of the language in its purest form.

3.5 Proof

We will now present our main result, with an outline proof. For a full proof the reader is referred to (Clark and Eyraud, 2005).

Theorem 1 This algorithm polynomially identifies in the limit the class of substitutable context-free languages.

Proof (Sketch) We can assume without loss of generality that the target grammar is in Chomsky normal form. We first define a characteristic set, that is to say a set of strings such that whenever the sample includes the characteristic set, the algorithm will output a correct grammar.

We define $w(\alpha) \in \Sigma^*$ to be the smallest word, according to $\prec$, generated by $\alpha \in (\Sigma \cup V)^+$. For each non-terminal $N \in V$ define $c(N)$ to be the smallest pair of terminal strings (l, r) (extending $\prec$ from Σ^* to $\Sigma^* \times \Sigma^*$, in some way), such that $S \stackrel{*}{\Rightarrow} lNr$.

We can now define the characteristic set $CS = \{lwr | (N \rightarrow \alpha) \in P, (l, r) = c(N), w = w(\alpha)\}$. The cardinality of this set is at most $|P|$ which is clearly polynomially bounded. We observe that the computations involved can all be polynomially bounded in the total size of the sample.

We next show that whenever the algorithm encounters a sample that includes this characteristic set, it outputs the right grammar. We write $\hat{G}$ for the learned grammar. Suppose $[u] \stackrel{*}{\Rightarrow}_{\hat{G}} v$. Then we can see that $u \equiv_L v$ by induction on the maximum length of the derivation of v. At each step we must use some rule $[u'] \Rightarrow [v'][w']$. It is easy to see that every rule of this type preserves the syntactic congruence of the left and right sides of the rules. Intuitively, the algorithm will never generate too large a language, since the languages are substitutable. Conversely, if we have a derivation of a string u with respect to the target grammar G, by

construction of the characteristic set, we will have, for every production $L \rightarrow MN$ in the target grammar, a production in the hypothesized grammar of the form $[w(L)] \rightarrow [w(M)][w(N)]$, and for every production of the form $L \rightarrow a$ we have a production $[w(L)] \rightarrow a$. A simple recursive argument shows that the hypothesized grammar will generate all the strings in the target language. Thus the grammar will generate all and only the strings required (QED).

3.6 Related work

This is the first provably correct and efficient grammatical inference algorithm for a linguistically interesting class of context-free grammars (but see for example (Yokomori, 2003) on the class of very simple grammars). It can also be compared to Angluin's famous work on reversible grammars (Angluin, 1982) which inspired a similar paper(Pilato and Berwick, 1985).

4 Experiments

We decided to see whether this algorithm without modification could shed some light on the debate discussed above. The experiments we present here are not intended to be an exhaustive test of the learnability of natural language. The focus is on determining whether learning can proceed in the absence of positive samples, and given only a very weak general purpose bias.

4.1 Implementation

We have implemented the algorithm described above. There are a number of algorithmic issues that were addressed. First, in order to find which pairs of strings are substitutable, the naive approach would be to compare strings pairwise which would be quadratic in the number of sentences. A more efficient approach maintains a hashtable mapping from contexts to congruence classes. Caching hashcodes, and using a union-find algorithm for merging classes allows an algorithm that is effectively linear in the number of sentences.

In order to handle large data sets with thousands of sentences, it was necessary to modify the algorithm in various ways which slightly altered its formal properties. However for the experiments reported here we used a version which performs exactly in line with the mathematical description above.

the man who is hungry died .
the man ordered dinner .
the man died .
the man is hungry .
is the man hungry ?
the man is ordering dinner .
is the man who is hungry ordering dinner ?
$*$is the man who hungry is ordering dinner ?

Table 1: Auxiliary fronting data set. Examples above the line were presented to the algorithm during the training phase, and it was tested on examples below the line.

4.2 Data

For clarity of exposition, we have used extremely small artificial data-sets, consisting only of sentences of types that would indubitably occur in the linguistic experience of a child.

Our first experiments were intended to determine whether the algorithm could determine the correct form of a polar question when the noun phrase had a relative clause, even when the algorithm was not exposed to any examples of that sort of sentence. We accordingly prepared a small data set shown in Table 1. Above the line is the training data that the algorithm was trained on. It was then tested on all of the sentences, including the ones below the line. By construction the algorithm would generate all sentences it has already seen, so it scores correctly on those. The learned grammar also correctly generated the correct form and did not generate the final form.

We can see how this happens quite easily since the simple nature of the algorithm allows a straightforward analysis. We can see that in the learned grammar "the man" will be congruent to "the man who is hungry", since there is a pair of sentences which differ only by this. Similarly, "hungry" will be congruent to "ordering dinner". Thus the sentence "is the man hungry ?" which is in the language, will be congruent to the correct sentence.

One of the derivations for this sentence would be: [is the man hungry ?] $\rightarrow$ [is the man hungry] [?] $\rightarrow$ [is the man] [hungry] [?] $\rightarrow$ [is] [the man] [hungry] [?] $\rightarrow$ [is] [the man][who is hungry] [hungry] [?] $\rightarrow$

it rains
it may rain
it may have rained
it may be raining
it has rained
it has been raining
it is raining
it may have been raining
∗it may have been rained
∗it may been have rain
∗it may have been rain

Table 2: English auxiliary data. Training data above the line, and testing data below.

[is] [the man][who is hungry] [ordering dinner] [?].

Our second data set is shown in Table 2, and is a fragment of the English auxiliary system. This has also been claimed to be evidence in favour of nativism. This was discussed in some detail by (Pilato and Berwick, 1985). Again the algorithm correctly learns.

5 Discussion

Chomsky was among the first to point out the limitations of Harris's approach, and it is certainly true that the grammars produced from these toy examples overgenerate radically. On more realistic language samples this algorithm would eventually start to generate even the incorrect forms of polar questions.

Given the solution we propose it is worth looking again and examining why nativists have felt that AFIPI was such an important issue. It appears that there are several different areas. First, the debate has always focussed on how to construct the interrogative from the declarative form. The problem has been cast as finding which auxilary should be "moved". Implicit in this is the assumption that the interrogative structure must be defined with reference to the declarative, one of the central assumptions of traditional transformational grammar. Now, of course, given our knowledge of many different formalisms which can correctly generate these forms without movement we can see that this assumption is false. There is of course a relation between these two sentences, a semantic one, but this does not imply that there need be any particular syntactic relation, and certainly not a "generative" relation.

Secondly, the view of learning algorithms is very narrow. It is considered that only sentences of that exact type could be relevant. We have demonstrated, if nothing else, that that view is false. The distinction can be learnt from a set of data that does not include any example of the exact piece of data required: as long as the various parts can be learned separately, the combination will function in the natural way.

A more interesting question is the extent to which the biases implicit in the learning algorithm are domain specific. Clearly the algorithm has a strong bias. It overgeneralises massively. One of the advantages of the algorithm for the purposes of this paper is that its triviality allows a remarkably clear and explicit statement of its bias. But is this bias specific to the domain of language? It in no way refers to anything specific to the field of language, still less specific to human language – no references to parts of speech, or phrases, or even hierarchical phrase structure. It is now widely recognised that this sort of recursive structure is domain-general (Jackendoff and Pinker, 2005).

We have selected for this demonstration an algorithm from grammatical inference. A number of statistical models have been proposed over the last few years by researchers such as (Klein and Manning, 2002; Klein and Manning, 2004) and (Solan et al., 2005). These models impressively manage to extract significant structure from raw data. However, for our purposes, neither of these models is suitable. Klein and Manning's model uses a variety of different cues, which combine with some specific initialisation and smoothing, and an explicit constraint to produce binary branching trees. Though very impressive, the model is replete with domain-specific biases and assumptions. Moreover, it does not learn a language in the strict sense (a subset of the set of all strings), though it would be a simple modification to make it perform such a task. The model by Solan et al. would be more suitable for this task, but again the complexity of the algorithm, which has numerous components and heuristics, and the lack of a theoretical justification for these heuristics again makes the task of identifying exactly what these biases are, and more importantly how domain specific they are,

a very significant problem.

In this model, the bias of the algorithm is completely encapsulated in the assumption $u \doteq v$ implies $u \equiv v$. It is worth pointing out that this does not even need hierarchical structure – the model could be implemented purely as a reduction system or semi-Thue system. The disadvantage of using that approach is that it is possible to construct some bizarre examples where the number of reductions can be exponential.

Using statistical properties of the set of strings, it is possible to extend these learnability results to a more substantial class of context free languages, though it is unlikely that these methods could be extended to a class that properly contains all natural languages.

6 Conclusion

We have presented an analysis of the argument that the acquisition of auxiliary fronting in polar interrogatives supports linguistic nativism. Using a very simple algorithm based on the ideas of Zellig Harris, with a simple domain-general heuristic, we show that the empirical question as to the frequency of occurrence of polar questions of a certain type in child-directed speech is a moot point, since the distinction in question can be learned even when no such sentences occur.

Acknowledgements This work has been partially supported by the EU funded PASCAL Network of Excellence on Pattern Analysis, Statistical Modelling and Computational Learning.

References

D. Angluin. 1982. Inference of reversible languages. *Communications of the ACM*, 29:741–765.

Noam Chomsky. 1975. *The Logical Structure of Linguistic Theory*. University of Chicago Press.

Alexander Clark and Remi Eyraud. 2005. Identification in the limit of substitutable context free languages. In Sanjay Jain, Hans Ulrich Simon, and Etsuji Tomita, editors, *Proceedings of The 16th International Conference on Algorithmic Learning Theory*, pages 283–296. Springer-Verlag.

S. Crain and M. Nakayama. 1987. Structure dependence in grammar formation. *Language*, 63(522-543).

C. de la Higuera. 1997. Characteristic sets for polynomial grammatical inference. *Machine Learning*, (27):125–138. Kluwer Academic Publishers. Manufactured in Netherland.

E. M. Gold. 1967. Language identification in the limit. *Information and control*, 10(5):447 – 474.

Zellig Harris. 1954. Distributional structure. *Word*, 10(2-3):146–62.

Ray Jackendoff and Steven Pinker. 2005. The nature of the language faculty and its implications for the evolution of language. *Cognition*, 97:211–225.

X. N. C. Kam, I. Stoyneshka, L. Tornyova, J. D. Fodor, and W. G. Sakas. 2005. Non-robustness of syntax acquisition from n-grams: A cross-linguistic perspective. In *The 18th Annual CUNY Sentence Processing Conference*, April.

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

Dan Klein and Chris Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the ACL*.

Samuel F. Pilato and Robert C. Berwick. 1985. Reversible automata and induction of the english auxiliary system. In *Proceedings of the ACL*, pages 70–75.

Geoffrey K. Pullum and Barbara C. Scholz. 2002. Empirical assessment of stimulus poverty arguments. *The Linguistic Review*, 19(1-2):9–50.

Florencia Reali and Morten H. Christiansen. 2004. Structure dependence in language acquisition: Uncovering the statistical richness of the stimulus. In *Proceedings of the 26th Annual Conference of the Cognitive Science Society*, Mahwah, NJ. Lawrence Erlbaum.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 354–362, Ann Arbor, Michigan, June.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. 2005. Unsupervised learning of natural languages. *Proc. Natl. Acad. Sci.*, 102:11629–11634.

Menno van Zaanen. 2000. ABL: Alignment-based learning. In *COLING 2000 - Proceedings of the 18th International Conference on Computational Linguistics*.

Takashi Yokomori. 2003. Polynomial-time identification of very simple grammars from positive data. *Theoretical Computer Science*, 298(1):179–206.

Using Gazetteers in Discriminative Information Extraction

Andrew Smith
Division of Informatics
University of Edinburgh
United Kingdom
a.p.smith-2@sms.ed.ac.uk

Miles Osborne
Division of Informatics
University of Edinburgh
United Kingdom
miles@inf.ed.ac.uk

Abstract

Much work on information extraction has successfully used gazetteers to recognise uncommon entities that cannot be reliably identified from local context alone. Approaches to such tasks often involve the use of maximum entropy-style models, where gazetteers usually appear as highly informative features in the model. Although such features can improve model accuracy, they can also introduce hidden negative effects. In this paper we describe and analyse these effects and suggest ways in which they may be overcome. In particular, we show that by quarantining gazetteer features and training them in a separate model, then decoding using a logarithmic opinion pool (Smith et al., 2005), we may achieve much higher accuracy. Finally, we suggest ways in which other features with gazetteer feature-like behaviour may be identified.

1 Introduction

In recent years discriminative probabilistic models have been successfully applied to a number of information extraction tasks in natural language processing (NLP), such as named entity recognition (NER) (McCallum and Li, 2003), noun phrase chunking (Sha and Pereira, 2003) and information extraction from research papers (Peng and McCallum, 2004). Discriminative models offer a significant advantage over their generative counterparts by allowing the specification of powerful, possibly non-independent features which would be difficult to tractably encode in a generative model.

In a task such as NER, one sometimes encounters an entity which is difficult to identify using local contextual cues alone because the entity has not be seen before. In these cases, a **gazetteer** or dictionary of possible entity identifiers is often useful. Such identifiers could be names of people, places, companies or other organisations. Using gazetteers one may define additional features in the model that represent the dependencies between a word's NER label and its presence in a particular gazetteer. Such gazetteer features are often highly informative, and their inclusion in the model should in principle result in higher model accuracy. However, these features can also introduce hidden negative effects taking the form of labelling errors that the model makes at places where a model without the gazetteer features would have labelled correctly. Consequently, ensuring optimal usage of gazetteers can be difficult.

In this paper we describe and analyse the labelling errors made by a model, and show that they generally result from the model's over-dependence on the gazetteer features for making labelling decisions. By including gazetteer features in the model we may, in some cases, transfer too much explanatory dependency to the gazetteer features from the non-gazetteer features. In order to avoid this problem, a more careful treatment of these features is required during training. We demonstrate that a traditional regularisation approach, where different features are regularised to different degrees, does not offer a sat-

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 133–140, New York City, June 2006.

isfactory solution. Instead, we show that by training gazetteer features in a separate model to the other features, and decoding using a **logarithmic opinion pool** (LOP) (Smith et al., 2005), much greater accuracy can be obtained. Finally, we identify other features with gazetteer feature-like properties and show that similar results may be obtained using our method with these features.

We take as our model a linear chain conditional random field (CRF), and apply it to NER in English.

2 Conditional Random Fields

A linear chain conditional random field (CRF) (Lafferty et al., 2001) defines the conditional probability of a label sequence **s** given an observed sequence **o** via:

$$p(\mathbf{s} \mid \mathbf{o}) = \frac{1}{Z(\mathbf{o})} \exp\left(\sum_{t=1}^{T+1} \sum_{k} \lambda_k f_k(s_{t-1}, s_t, \mathbf{o}, t)\right) \quad (1)$$

where T is the length of both sequences, λ_k are parameters of the model and $Z(\mathbf{o})$ is a partition function that ensures that (1) represents a probability distribution. The functions f_k are feature functions representing the occurrence of different events in the sequences **s** and **o**.

The parameters λ_k can be estimated by maximising the conditional log-likelihood of a set of labelled training sequences. At the maximum likelihood solution the model satisfies a set of feature constraints, whereby the expected count of each feature under the model is equal to its empirical count on the training data:

$$E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] = 0, \ \forall k$$

In general this cannot be solved for the λ_k in closed form, so numerical optimisation must be used. For our experiments we use the limited memory variable metric (LMVM) (Sha and Pereira, 2003) routine, which has become the standard algorithm for CRF training with a likelihood-based objective function.

To avoid overfitting, a prior distribution over the model parameters is typically used. A common example of this is the Gaussian prior. Use of a prior involves adding extra terms to the objective and its derivative. In the case of a Gaussian prior, these additional terms involve the mean and variance of the distribution.

3 Previous Use of Gazetteers

Gazetteers have been widely used in a variety of information extraction systems, including both rule-based systems and statistical models. In addition to lists of people names, locations, etc., recent work in the biomedical domain has utilised gazetteers of biological and genetic entities such as gene names (Finkel et al., 2005; McDonald and Pereira, 2005). In general gazetteers are thought to provide a useful source of external knowledge that is helpful when an entity cannot be identified from knowledge contained solely within the data set used for training. However, some research has questioned the usefulness of gazetteers (Krupka and Hausman, 1998). Other work has supported the use of gazetteers in general but has found that lists of only moderate size are sufficient to provide most of the benefit (Mikheev et al., 1999). Therefore, to date the effective use of gazetteers for information extraction has in general been regarded as a "black art". In this paper we explain some of the likely reasons for these findings, and propose ways to more effectively handle gazetteers when they are used by maxent-style models.

In work developed independently and in parallel to the work presented here, Sutton et al. (2006) identify general problems with gazetteer features and propose a solution similar to ours. They present results on NP-chunking in addition to NER, and provide a slightly more general approach. By contrast, we motivate the problem more thoroughly through analysis of the actual errors observed and through consideration of the success of other candidate solutions, such as traditional regularisation over feature subsets.

4 Our Experiments

In this section we describe our experimental setup, and provide results for the baseline models.

4.1 Task and Dataset

Named entity recognition (NER) involves the identification of the location and type of pre-defined entities within a sentence. The CRF is presented with a set of sentences and must label each word so as to indicate whether the word appears outside an entity, at the beginning of an entity of a certain type or

within the continuation of an entity of a certain type.

Our results are reported on the CoNLL-2003 shared task English dataset (Sang and Meulder, 2003). For this dataset the entity types are: persons (PER), locations (LOC), organisations (ORG) and miscellaneous (MISC). The training set consists of 14,987 sentences and 204,567 tokens, the development set consists of 3,466 sentences and 51,578 tokens and the test set consists of 3,684 sentences and 46,666 tokens.

4.2 Gazetteers

We employ a total of seven gazetteers for our experiments. These cover names of people, places and organisations. Specifically, we have gazetteers containing surnames (88,799 entries), female first names (4,275 entries), male first names (1,219 entries), names of places (27,635 entries), names of companies (20,638 and 279,195 entries) and names of other organisations (425 entries).

4.3 Feature set

Our experiments are centred around two CRF models, one with and one without gazetteer features. The model *without* gazetteer features, which we call **standard**, comprises features defined in a window of five words around the current word. These include features encoding *n*-grams of words and POS tags, and features encoding orthographic properties of the current word. The orthographic features are based on those found in (Curran and Clark, 2003). Examples include whether the current word is capitalised, is an initial, contains a digit, contains punctuation, etc. In total there are 450,345 features in the **standard** model.

We call the second model, *with* gazetteer features, **standard+g**. This includes all the features contained in the **standard** model as well as 8,329 gazetteer features. Our gazetteer features are a typical way to represent gazetteer information in maxent-style models. They are divided into two categories: *unlexicalised* and *lexicalised*. The unlexicalised features model the dependency between a word's presence in a gazetteer and its NER label, irrespective of the word's identity. The lexicalised features, on the other hand, include the word's identity and so provide more refined word-specific modelling of the gazetteer-NER label dependency.[1] There are 35 unlexicalised gazetteer features and 8,294 lexicalised gazetteer features, giving a total of 458,675 features in the **standard+g** model.

Model	Development		Test	
	Unreg.	Reg.	Unreg.	Reg.
standard	88.21	89.86	81.60	83.97
standard+g	89.19	90.40	83.10	84.70

Table 1: Model F scores

		standard+g ✓	standard+g ✗
standard	✓	44,945	160
standard	✗	228	1,333

Table 2: Test set errors

4.4 Baseline Results

Table 1 gives F scores for the **standard** and **standard+g** models. Development set scores are included for completeness, and are referred to later in the paper. We show results for both unregularised and regularised models. The regularised models are trained with a zero-mean Gaussian prior, with the variance set using the development data.

We see that, as expected, the presence of the gazetteer features allows **standard+g** to outperform **standard**, for both the unregularised and regularised models. To test significance, we use McNemar's matched-pairs test (Gillick and Cox, 1989) on pointwise labelling errors. In each case, the **standard+g** model outperforms the **standard** model at a significance level of $p < 0.02$. However, these results camouflage the fact that the gazetteer features introduce some negative effects, which we explore in the next section. As such, the real benefit of including the gazetteer features in **standard+g** is not fully realised.

5 Problems with Gazetteer Features

We identify problems with the use of gazetteer features by considering test set labelling errors for both **standard** and **standard+g**. We use regularised models here as an illustration. Table 2 shows the

[1] Many gazetteer entries involve strings of words where the individual words in the string do not appear in the gazetteer in isolation. For this reason the lexicalised gazetteer features are *not* simply determined by the word identity features.

number of sites (a site being a particular word at a particular position in a sentence) where labellings have improved, worsened or remained unchanged with respect to the gold-standard labelling with the addition of the gazetteer features. For example, the value in the top-left cell is the number of sites where both the **standard** and **standard+g** label words correctly.

The most interesting cell in the table is the top-right one, which represents sites where **standard** is correctly labelling words but, with the addition of the gazetteer features, **standard+g** mislabels them. At these sites, the addition of the gazetteer features actually worsens things. How well, then, could the **standard+g** model do if it could somehow reduce the number of errors in the top-right cell? In fact, if it had correctly labelled those sites, a significantly higher test set F score of 90.36% would have been obtained. This potential upside suggests much could be gained from investigating ways of correcting the errors in the top-right cell. It is not clear whether there exists any approach that could correct *all* the errors in the top-right cell while simultaneously maintaining the state in the other cells, but approaches that are able to correct at least some of the errors should prove worthwhile.

On inspection of the sites where errors in the top-right cell occur, we observe that some of the errors occur in sequences where no words are in any gazetteer, so no gazetteer features are active for any possible labelling of these sequences. In other cases, the errors occur at sites where some of the gazetteer features appear to have dictated the label, but have made an incorrect decision. As a result of these observations, we classify the errors from the top-right cell of Table 2 into two types: *type A* and *type B*.

5.1 Type A Errors

We call type A errors those errors that occur at sites where gazetteer features seem to have been *directly* responsible for the mislabelling. In these cases the gazetteer features effectively "over-rule" the other features in the model causing a mislabelling where the **standard** model, without the gazetteer features, correctly labels the word.

An example of a type A error is given in the sentence extract below:

```
about/O Healy/I-LOC
```

This is the labelling given by **standard+g**. The correct label for `Healy` here is `I-PER`. The **standard** model is able to decode this correctly as `Healy` appears in the training data with the `I-PER` label. The reason for the mislabelling by the **standard+g** model is that `Healy` appears in both the gazetteer of place names and the gazetteer of person surnames. The feature encoding the gazetteer of place names with the `I-LOC` label has a λ value of 4.20, while the feature encoding the gazetteer of surnames with the `I-PER` label has a λ value of 1.96, and the feature encoding the word `Healy` with the `I-PER` label has a λ value of 0.25. Although other features both at the word `Healy` and at other sites in the sentence contribute to the labelling of `Healy`, the influence of the first feature above dominates. So in this case the addition of the gazetteer features has confused things.

5.2 Type B Errors

We call type B errors those errors that occur at sites where the gazetteer features seem to have been only *indirectly* responsible for the mislabelling. In these cases the mislabelling appears to be more attributable to the non-gazetteer features, which are in some sense less expressive after being trained with the gazetteer features. Consequently, they are less able to decode words that they could previously label correctly.

An example of a type B error is given in the sentence extract below:

```
Chanderpaul/O was/O
```

This is the labelling given by **standard+g**. The correct labelling, given by **standard**, is `I-PER` for `Chanderpaul`. In this case no words in the sentence (including the part not shown) are present in any of the gazetteers so no gazetteer features are active for any labelling of the sentence. Consequently, the gazetteer features do not contribute at all to the labelling decision. Non-gazetteer features in **standard+g** are, however, unable to find the correct labelling for `Chanderpaul` when they previously could in the **standard** model.

For both type A and type B errors it is clear that the gazetteer features in **standard+g** are in some

sense too "powerful" while the non-gazetteers features have become too "weak". The question, then, is: can we train all the features in the model in a more sophisticated way so as to correct for these effects?

6 Feature Dependent Regularisation

One interpretation of the findings of our error analysis above is that the addition of the gazetteer features to the model is having an implicit over-regularising effect on the other features. Therefore, is it possible to adjust for this effect through more careful explicit regularisation using a prior? Can we directly regularise the gazetteer features more heavily and the non-gazetteer features less? We investigate this possibility in this section.

The **standard+g** model is regularised by fitting a single Gaussian variance hyperparameter across all features. The optimal value for this single hyperparameter is 45. We now relax this single constraint by allocating a separate variance hyperparameter to different feature subsets, one for the gazetteer features (σ_{gaz}) and one for the non-gazetteer features ($\sigma_{non\text{-}gaz}$). The hope is that the differing subsets of features are best regularised using different prior hyperparameters. This is a natural approach within most standardly formulated priors for log-linear models. Clearly, by doing this we increase the search space significantly. In order to make the search manageable, we constrain ourselves to three scenarios: (1) Hold $\sigma_{non\text{-}gaz}$ at 45, and regularise the gazetteer features a little more by reducing σ_{gaz}. (2) Hold σ_{gaz} at 45, and regularise the non-gazetteer features a little less by increasing $\sigma_{non\text{-}gaz}$. (3) Simultaneously regularise the gazetteer features a little more than at the single variance optimum, and regularise the non-gazetteer features a little less.

Table 3 gives representative development set F scores for each of these three scenarios, with each scenario separated by a horizontal dividing line. We see that in general the results do not differ significantly from that of the single variance optimum. We conjecture that the reason for this is that the regularising effect of the gazetteer features on the non-gazetteer features is due to relatively subtle interactions during training that relate to the dependencies the features encode and how these dependencies overlap. Regularising different feature subsets by different amounts with a Gaussian prior does not directly address these interactions but instead just rather crudely penalises the magnitude of the parameter values of different feature sets to different degrees. Indeed this is true for any standardly formulated prior. It seems therefore that any solution to the regularising problem should come through more explicit restricting or removing of the interactions between gazetteer and non-gazetteer features during training.

σ_{gaz}	$\sigma_{non-gaz}$	F score
42	45	90.40
40	45	90.30
45	46	90.39
45	50	90.38
44.8	45.2	90.41
43	47	90.35

Table 3: FDR development set F scores

7 Combining Separately Trained Models

We may remove interactions between gazetteer and non-gazetteer features entirely by quarantining the gazetteer features and training them in a separate model. This allows the non-gazetteer features to be protected from the over-regularising effect of the gazetteer features. In order to decode taking advantage of the information contained in both models, we must combine the models in some way. To do this we use a **logarithmic opinion pool** (LOP) (Smith et al., 2005). This is similar to a mixture model, but uses a weighted multiplicative combination of models rather than a weighted additive combination. Given models p_α and per-model weights w_α, the LOP distribution is defined by:

$$p_{\text{LOP}}(\mathbf{s} \mid \mathbf{o}) = \frac{1}{Z_{\text{LOP}}(\mathbf{o})} \prod_\alpha [p_\alpha(\mathbf{s} \mid \mathbf{o})]^{w_\alpha} \qquad (2)$$

with $w_\alpha \geq 0$ and $\sum_\alpha w_\alpha = 1$, and where $Z_{\text{LOP}}(\mathbf{o})$ is a normalising function. The weight w_α encodes the dependence of the LOP on model α. In the case of a CRF, the LOP itself is a CRF and so decoding is no more complex than for standard CRF decoding.

In order to use a LOP for decoding we must set the weights w_α in the weighted product. In (Smith et

Feature Subset	Feature Type
s1	simple structural features
s2	advanced structural features
n	*n*-grams of words and POS tags
o	simple orthographic features
a	advanced orthographic features
g	gazetteer features

Table 4: **standard+g** feature subsets

LOP	Dev Set	Test Set
standard+g	90.40	84.70
s1g-standard	**91.34**	**85.98**
s2g-standard	91.32	85.59
s2ng-standard	90.66	84.59
s2nog-standard	90.47	84.92
s2noag-standard	90.56	84.78

Table 5: Reg. LOP F scores

LOP	LOP Weights
s1g-standard	[0.39, 0.61]
s2g-standard	[0.29, 0.71]
s2ng-standard	[0.43, 0.57]
s2nog-standard	[0.33, 0.67]
s2noag-standard	[0.39, 0.61]

Table 6: Reg. LOP weights

al., 2005) a procedure is described whereby the (normalised) weights are explicitly trained. In this paper, however, we only construct LOPs consisting of two models in each case, one model with gazetteer features and one without. We therefore do not require the weight training procedure as we can easily fit the two weights (only one of which is free) using the development set.

To construct models for the gazetteer and non-gazetteer features we first partition the feature set of the **standard+g** model into the subsets outlined in Table 4. The *simple structural features* model label-label and label-word dependencies, while the *advanced structural features* include these features as well as those modelling label-label-word conjunctions. The *simple orthographic features* measure properties of a word such as capitalisation, presence of a digit, etc., while the *advanced orthographic properties* model the occurrence of prefixes and suffixes of varying length.

We create and train different models for the gazetteer features by adding different feature subsets to the gazetteer features. We regularise these models in the usual way using a Gaussian prior. In each case we then combine these models with the **standard** model and decode under a LOP.

Table 5 gives results for LOP decoding for the different model pairs. Results for the **standard+g** model are included in the first row for comparison. For each LOP the hyphen separates the two models comprising the LOP. So, for example, in the second row of the table we combine the gazetteer features with simple structural features in a model, train and decode with the **standard** model using a LOP. The simple structural features are included so as to provide some basic support to the gazetteer features.

We see from Table 5 that the first two LOPs significantly outperform the regularised **standard+g** model (at a significance level of $p < 0.01$, on both the test and development sets). By training the gazetteer features separately we have avoided their over-regularising effect on the non-gazetteer features. This relies on training the gazetteer features with a relatively small set of other features. This is illustrated as we read down the table, below the top two rows. As more features are added to the model containing the gazetteer features we obtain decreasing test set F scores because the advantage created from separate training of the features is increasingly lost.

Table 6 gives the corresponding weights for the LOPs in Table 5, which are set using the development data. We see that in every case the LOP allocates a smaller weight to the gazetteer features model than the non-gazetteer features model and in doing so restricts the influence that the gazetteer features have in the LOP's labelling decisions.

Table 7, similar to Table 2 earlier, shows test set labelling errors for the **standard** model and one of the LOPs. We take the **s2g-standard** LOP here for illustration. We see from the table that the number of errors in the top-right cell shows a reduction of 29% over the corresponding value in Table 2. We have therefore reduced the number errors of the type we were targeting with our approach. The approach has also had the effect of reducing the number of errors in the bottom-right cell, which further improves model accuracy.

All the LOPs in Table 5 contain regularised mod-

standard	s2g-standard LOP ✓	✗
✓	44,991	114
✗	305	1,256

Table 7: Test set errors

LOP	Dev Set	Test Set
s1g-standard	**90.58**	**84.87**
s2g-standard	90.70	84.28
s2ng-standard	89.70	84.01
s2nog-standard	89.48	83.99
s2noag-standard	89.40	83.70

Table 8: Unreg. LOP F scores

els. Table 8 gives test set F scores for the corresponding LOPs constructed from unregularised models. As we would expect, the scores are lower than those in Table 5. However, it is interesting to note that the **s1g-standard** LOP still outperforms the *regularised* **standard+g** model.

In summary, by training the gazetteer features and non-gazetteer features in separate models and decoding using a LOP, we are able to overcome the problems described in earlier sections and can achieve much higher accuracy. This shows that successfully deploying gazetteer features within maxent-style models should involve careful consideration of restrictions on how features interact with each other, rather than simply considering the absolute values of feature parameters.

8 Gazetteer-Like Features

So far our discussion has focused on gazetteer features. However, we would expect that the problems we have described and dealt with in the last section also occur with other types of features that have similar properties to gazetteer features. By applying similar treatment to these features during training we may be able harness their usefulness to a greater degree than is currently the case when training in a single model. So how can we identify these features?

The task of identifying the optimal partitioning for creation of models in the previous section is in general a hard problem as it relies on clustering the features based on their explanatory power relative to all other clusters. It may be possible, however, to devise some heuristics that approximately correspond to the salient properties of gazetteer features (with respect to the clustering) and which can then be used to identify other features that have these properties. In this section we consider three such heuristics. All of these heuristics are motivated by the observation that gazetteer features are both highly discriminative and generally very sparse.

Family Singleton Features We define a feature *family* as a set of features that have the same conjunction of predicates defined on the observations. Hence they differ from each other only in the NER label that they encode. *Family singleton features* are features that have a count of 1 in the training data when all other members of that feature family have zero counts. These features have a flavour of gazetteer features in that they represent the fact that the conjunction of observation predicates they encode is highly predictive of the corresponding NER label, and that they are also very sparse.

Family *n*-ton Features These are features that have a count of n (greater than 1) in the training data when all other members of that feature family have zero counts. They are similar to family singleton features, but exhibit gazetteer-like properties less and less as the value of n is increased because a larger value of n represents less sparsity.

Loner Features These are features which occur with a low mean number of other features in the training data. They are similar to gazetteer features in that, at the points where they occur, they are in some sense being relied upon more than most features to explain the data. To create loner feature sets we rank all features in the **standard+g** model based on the mean number of other features they are observed with in the training data, then we take subsets of increasing size. We present results for subsets of size 500, 1000, 5000 and 10000.

For each of these categories of features we add simple structural features (the **s1** set from earlier), to provide basic structural support, and then train a regularised model. We also train a regularised model consisting of all features in **standard+g** except the features from the category in question. We decode these model pairs under a LOP as described earlier.

Table 9 gives test set F scores for LOPs created from each of the categories of features above

LOP	Test Set
FSF	85.79
FnF	84.78
LF 500	85.80
LF 1000	85.70
LF 5000	85.77
LF 10000	85.62

Table 9: Reg. LOP F scores

(with abbreviated names derived from the category names). The results show that for the *family singleton features* and each of the *loner feature* sets we obtain LOPs that significantly outperform the regularised **standard+g** model ($p < 0.0002$ in every case). The *family n-ton features'* LOP does not do as well, but that is probably due to the fact that some of the features in this set have a large value of *n* and so behave much less like gazetteer features.

In summary, we obtain the same pattern of results using our quarantined training and LOP decoding method with these categories of features that we do with the gazetteer features. We conclude that the problems with gazetteer features that we have identified in this paper are exhibited by general discriminative features with gazetteer feature-like properties, and our method is also successful with these more general features. Clearly, the heuristics that we have devised in this section are very simple, and it is likely that with more careful engineering better feature partitions can be found.

9 Conclusion and future work

In this paper we have identified and analysed negative effects that can be introduced to maxent-style models by the inclusion of highly discriminative gazetteer features. We have shown that such effects manifest themselves through errors that generally result from the model's over-dependence on the gazetteer features for decision making. To overcome this problem a more careful treatment of these features is required during training. We have proposed a solution that involves quarantining the features and training them separately to the other features in the model, then decoding the separate models with a logarithmic opinion pool. In fact, the LOP provides a natural way to handle the problem, with different constituent models for the different feature types. The method leads to much greater accuracy, and allows the power of gazetteer features to be more effectively harnessed. Finally, we have identified other feature sets with gazetteer feature-like properties and shown that similar results may be obtained using our method with these feature sets.

In this paper we defined intuitively-motivated feature partitions (gazetteer feature-based or otherwise) using heuristics. In future work we will focus on automatically determining such partitions.

References

James Curran and Stephen Clark. 2003. Language independent NER using a maximum entropy tagger. In *Proc. CoNLL-2003*.

Jenny Finkel, Shipra Dingare, Christopher D. Manning, Malvina Nissim, Beatrice Alex, and Claire Grover. 2005. Exploring the boundaries: gene and protein identification in biomedical text. *BMC Bioinformatics*, (6).

L. Gillick and Stephen Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 532–535.

George R. Krupka and Kevin Hausman. 1998. Isoquest Inc: Description of the NetOwl (TM) extractor system as used for MUC-7. In *Proc. MUC-7*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML 2001*.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proc. CoNLL-2003*.

Ryan McDonald and Fernando Pereira. 2005. Identifying gene and protein mentions in text using conditional random fields. *BMC Bioinformatics*, (6).

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers.

Fuchun Peng and Andrew McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In *Proc. HLT-NAACL 2004*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. CoNLL-2003*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proc. HLT-NAACL 2003*.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *Proc. ACL 2005*.

Charles Sutton, Michael Sindelar, and Andrew McCallum. 2006. Reducing weight undertraining in struxctured discriminative learning. In *Proc. HLT/NAACL 2006*.

A Context Pattern Induction Method for Named Entity Extraction

Partha Pratim Talukdar
CIS Department
University of Pennsylvania
Philadelphia, PA 19104
partha@cis.upenn.edu

Thorsten Brants
Google, Inc.
1600 Amphitheatre Pkwy.
Mountain View, CA 94043
brants@google.com

Mark Liberman Fernando Pereira
CIS Department
University of Pennsylvania
Philadelphia, PA 19104
{myl,pereira}@cis.upenn.edu

Abstract

We present a novel context pattern induction method for information extraction, specifically named entity extraction. Using this method, we extended several classes of seed entity lists into much larger high-precision lists. Using token membership in these extended lists as additional features, we improved the accuracy of a conditional random field-based named entity tagger. In contrast, features derived from the seed lists decreased extractor accuracy.

1 Introduction

Partial entity lists and massive amounts of unlabeled data are becoming available with the growth of the Web as well as the increased availability of specialized corpora and entity lists. For example, the primary public resource for biomedical research, MEDLINE, contains over 13 million entries and is growing at an accelerating rate. Combined with these large corpora, the recent availability of entity lists in those domains has opened up interesting opportunities and challenges. Such lists are never complete and suffer from sampling biases, but we would like to exploit them, in combination with large unlabeled corpora, to speed up the creation of information extraction systems for different domains and languages. In this paper, we concentrate on exploring utility of such resources for named entity extraction.

Currently available entity lists contain a small fraction of named entities, but there are orders of magnitude more present in the unlabeled data[1]. In this paper, we test the following hypotheses:

i. Starting with a few seed entities, it is possible to induce high-precision context patterns by exploiting entity context redundancy.

ii. New entity instances of the same category can be extracted from unlabeled data with the induced patterns to create high-precision extensions of the seed lists.

iii. Features derived from token membership in the extended lists improve the accuracy of learned named-entity taggers.

Previous approaches to context pattern induction were described by Riloff and Jones (1999), Agichtein and Gravano (2000), Thelen and Riloff (2002), Lin et al. (2003), and Etzioni et al. (2005), among others. The main advance in the present method is the combination of grammatical induction and statistical techniques to create high-precision patterns.

The paper is organized as follows. Section 2 describes our pattern induction algorithm. Section 3 shows how to extend seed sets with entities extracted by the patterns from unlabeled data. Section 4 gives experimental results, and Section 5 compares our method with previous work.

[1] For example, based on approximate matching, there is an overlap of only 22 organizations between the 2403 organizations present in CoNLL-2003 shared task training data and the Fortune-500 list.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 141–148, New York City, June 2006.

2 Context Pattern Induction

The overall method for inducing entity context patterns and extending entity lists is as follows:

1. Let E = seed set, T = text corpus.
2. Find the contexts C of entities in E in the corpus T (Section 2.1).
3. Select *trigger words* from C (Section 2.2).
4. For each trigger word, induce a pattern automaton (Section 2.3).
5. Use induced patterns P to extract more entities E' (Section 3).
6. Rank P and E' (Section 3.1).
7. If needed, add high scoring entities in E' to E and return to step 2. Otherwise, terminate with patterns P and extended entity list $E \cup E'$ as results.

2.1 Extracting Context

Starting with the seed list, we first find occurrences of seed entities in the unlabeled data. For each such occurrence, we extract a fixed number W (context window size) of tokens immediately preceding and immediately following the matched entity. As we are only interested in modeling the context here, we replace all entity tokens by the single token `-ENT-`. This token now represents a *slot* in which an entity can occur. Examples of extracted entity contexts are shown in Table 1. In the work presented in this papers, seeds are entity instances (e.g. *Google* is a seed for organization category).

increased expression of `-ENT-` in vad mice
the expression of `-ENT-` mrna was greater
expression of the `-ENT-` gene in mouse

Table 1: Extracted contexts of known genes with $W = 3$.

The set of extracted contexts is denoted by C. The next step is to automatically induce high-precision patterns containing the token `-ENT-` from such extracted contexts.

2.2 Trigger Word Selection

To induce patterns, we need to determine their starts. It is reasonable to assume that some tokens are more specific to particular entity classes than others. For example, in the examples shown above, *expression* can be one such word for gene names. Whenever one comes across such a token in text, the probability of finding an entity (of the corresponding entity class) in its vicinity is high. We call such starting tokens *trigger words*. Trigger words mark the beginning of a pattern. It is important to note that simply selecting the first token of extracted contexts may not be a good way to select trigger words. In such a scheme, we would have to vary W to search for useful pattern starts. Instead of that brute-force technique, we propose an automatic way of selecting trigger words. A good set of trigger words is very important for the quality of induced patterns. Ideally, we want a trigger word to satisfy the following:

- It is frequent in the set C of extracted contexts.
- It is specific to entities of interest and thereby to extracted contexts.

We use a term-weighting method to rank candidate trigger words from entity contexts. IDF (Inverse Document Frequency) was used in our experiments but any other suitable term-weighting scheme may work comparably. The IDF weight f_w for a word w occurring in a corpus is given by:

$$f_w = \log\left(\frac{N}{n_w}\right)$$

where N is the total number of documents in the corpus and n_w is the total number of documents containing w. Now, for each context segment $c \in C$, we select a *dominating word* d_c given by

$$d_c = \arg\max_{w \in c} f_w$$

There is exactly one dominating word for each $c \in C$. All dominating words for contexts in C form multiset M. Let m_w be the multiplicity of the dominating word w in M. We sort M by decreasing m_w and select the top n tokens from this list as potential trigger words.

Selection criteria based on dominating word frequency work better than criteria based on simple term weight because high term weight words may be rare in the extracted contexts, but would still be misleadingly selected for pattern induction. This can be avoided by using instead the frequency of dominating words within contexts, as we did here.

2.3 Automata Induction

Rather than using individual contexts directly, we summarize them into automata that contain the most significant regularities of the contexts sharing a given trigger word. This construction allows us to determine the relative importance of different context features using a variant of the forward-backward algorithm from HMMs.

2.3.1 Initial Induction

For each trigger word, we list the contexts starting with the word. For example, with *"expression"* as the trigger word, the contexts in Table 1 are reduced to those in Table 2. Since *"expression"* is a left-context trigger word, only one token to the right of `-ENT-` is retained. Here, the predictive context lies to the left of the slot `-ENT-` and a single token is retained on the right to mark the slot's right boundary. To model predictive right contexts, the token string can be reversed and the same techniques as here applied on the reversed string.[2]

expression of `-ENT-` *in*
expression of `-ENT-` *mrna*
expression of the `-ENT-` *gene*

Table 2: Context segments corresponding to trigger word *"expression"*.

Similar contexts are prepared for each trigger word. The context set for each trigger word is then summarized by a pattern automaton with transitions that match the trigger word and also the wildcard `-ENT-` . We expect such automata to model the position in context of the entity slot and help us extract more entities of the same class with high precision.

[2]Experiments reported in this paper use predictive left context only.

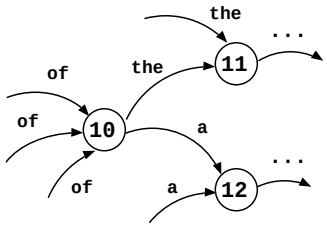


Figure 1: Fragment of a 1-reversible automaton

We use a simple form of grammar induction to learn the pattern automata. Grammar induction techniques have been previously explored for information extraction (IE) and related tasks. For instance, Freitag (1997) used grammatical inference to improve precision in IE tasks.

Context segments are short and typically do not involve recursive structures. Therefore, we chose to use 1-reversible automata to represent sets of contexts. An automaton A is *k-reversible* iff (1) A is deterministic and (2) A^r is deterministic with k tokens of lookahead, where A^r is the automaton obtained by reversing the transitions of A. Wrapper induction using *k-reversible* grammar is discussed by Chidlovskii (2000).

In the 1-reversible automaton induced for each trigger word, all transitions labeled by a given token go to the same state, which is identified with that token. Figure 1 shows a fragment of a 1-reversible automaton. Solan et al. (2005) describe a similar automaton construction, but they allow multiple transitions between states to distinguish among sentences.

Each transition $e = (v, w)$ in a 1-reversible automaton A corresponds to a bigram vw in the contexts used to create A. We thus assign each transition the probability

$$P(w|v) = \frac{C(v,w)}{\Sigma_{w'} C(v,w')}$$

where $C(v, w)$ is the number of occurrences of the bigram vw in contexts for W. With this construction, we ensure words will be credited in proportion to their frequency in contexts. The automaton may overgenerate, but that potentially helps generalization.

2.3.2 Pruning

The initially induced automata need to be pruned to remove transitions with weak evidence so as to increase match precision.

The simplest pruning method is to set a count threshold c below which transitions are removed. However, this is a poor method. Consider state 10 in the automaton of Figure 2, with $c = 20$. Transitions $(10, 11)$ and $(10, 12)$ will be pruned. $C(10, 12) \ll c$ but $C(10, 11)$ just falls short of c. However, from the transition counts, it looks like the sequence *"the -ENT-"* is very common. In such a case, it is not desirable to prune $(10, 11)$. Using a local threshold may lead to overpruning.

We would like instead to keep transitions that are used in relatively many probable paths through the automaton. The probability of path p is $P(p) = \prod_{(v,w)\in p} P(w|v)$. Then the posterior probability of edge (v, w) is

$$P(v, w) = \frac{\sum_{(v,w)\in p} P(p)}{\sum_p P(p)} \quad ,$$

which can be efficiently computed by the forward-backward algorithm (Rabiner, 1989). We can now remove transitions leaving state v whose posterior probability is lower than $p_v = k(\max_w P(v, w))$, where $0 < k \leq 1$ controls the degree of pruning, with higher k forcing more pruning. All induced and pruned automata are trimmed to remove unreachable states.

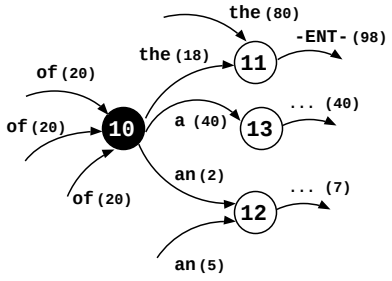


Figure 2: Automaton to be pruned at state 10. Transition counts are shown in parenthesis.

3 Automata as Extractor

Each automaton induced using the method described in Sections 2.3-2.3.2 represents high-precision patterns that start with a given trigger word. By scanning unlabeled data using these patterns, we can extract text segments which can be substituted for the slot token `-ENT-`. For example, assume that the induced pattern is *"analyst at -ENT- and"* and that the scanned text is *"He is an analyst at the University of California and ..."*. By scanning this text using the pattern mentioned above, we can figure out that the text *"the University of California"* can substitute for "`-ENT-`". This extracted segment is a candidate extracted entity. We now need to decide whether we should retain all tokens inside a candidate extraction or purge some tokens, such as *"the"* in the example.

One way to handle this problem is to build a language model of content tokens and retain only the maximum likelihood token sequence. However, in the current work, the following heuristic which worked well in practice is used. Each token in the extracted text segment is labeled either *keep* (`K`) or *droppable* (`D`). By default, a token is labeled `K`. A token is labeled `D` if it satisfies one of the droppable criteria. In the experiments reported in this paper, droppable criteria were whether the token is present in a stopword list, whether it is non-capitalized, or whether it is a number.

Once tokens in a candidate extraction are labeled using the above heuristic, the longest token sequence corresponding to the regular expression $\mathtt{K}[\mathtt{D}\ \mathtt{K}]*\mathtt{K}$ is retained and is considered a final extraction. If there is only one `K` token, that token is retained as the final extraction. In the example above, the tokens are labeled *"the/D University/K of/D California/K"*, and the extracted entity will be *"University of California"*.

To handle run-away extractions, we can set a domain-dependent hard limit on the number of tokens which can be matched with "`-ENT-`". This stems from the intuition that useful extractions are not very long. For example, it is rare that a person name longer than five tokens.

3.1 Ranking Patterns and Entities

Using the method described above, patterns and the entities extracted by them from unlabeled data are paired. But both patterns and extractions vary in quality, so we need a method for ranking both. Hence, we need to rank both patterns and entities. This is difficult given that there we have no nega-

tive labeled data. Seed entities are the only positive instances that are available.

Related previous work tried to address this problem. Agichtein and Gravano (2000) seek to extract relations, so their pattern evaluation strategy considers one of the attributes of an extracted tuple as a key. They judge the tuple as a positive or a negative match for the pattern depending on whether there are other extracted values associated with the same key. Unfortunately, this method is not applicable to entity extraction.

The pattern evaluation mechanism used here is similar in spirit to those of Etzioni et al. (2005) and Lin et al. (2003). With seeds for multiple classes available, we consider seed instances of one class as negative instances for the other classes. A pattern is penalized if it extracts entities which belong to the seed lists of the other classes. Let $\mathrm{pos}(p)$ and $\mathrm{neg}(p)$ be respectively the number of distinct positive and negative seeds extracted by pattern p. In contrast to previous work mentioned above, we do not combine $\mathrm{pos}(p)$ and $\mathrm{neg}(p)$ to calculate a single accuracy value. Instead, we discard all patterns p with positive $\mathrm{neg}(p)$ value, as well as patterns whose total positive seed (distinct) extraction count is less than certain threshold η_{pattern}. This scoring is very conservative. There are several motivations for such a conservative scoring. First, we are more interested in precision than recall. We believe that with massive corpora, large number of entity instances can be extracted anyway. High accuracy extractions allow us to reliably (without any human evaluation) use extracted entities in subsequent tasks successfully (see Section 4.3). Second, in the absence of sophisticated pattern evaluation schemes (which we are investigating — Section 6), we feel it is best to heavily penalize any pattern that extracts even a single negative instance.

Let G be the set of patterns which are retained by the filtering scheme described above. Also, let $I(e, p)$ be an indicator function which takes value 1 when entity *e* is extracted by pattern *p* and 0 otherwise. The score of e, $S(e)$, is given by

$$S(e) = \Sigma_{p \in G} I(e, p)$$

This whole process can be iterated by including extracted entities whose score is greater than or equal to a certain threshold η_{entity} to the seed list.

4 Experimental Results

For the experiments described below, we used 18 billion tokens (31 million documents) of news data as the source of unlabeled data. We experimented with 500 and 1000 trigger words. The results presented were obtained after a single iteration of the Context Pattern Induction algorithm (Section 2).

4.1 English LOC, ORG and PER

For this experiment, we used as seed sets subsets of the entity lists provided with CoNLL-2003 shared task data.[3] Only multi-token entries were included in the seed lists of respective categories (location (LOC), person (PER) & organization (ORG) in this case). This was done to partially avoid incorrect context extraction. For example, if the seed entity is *"California"*, then the same string present in *"University of California"* can be incorrectly considered as an instance of LOC. A stoplist was used for dropping tokens from candidate extractions, as described in Section 3. Examples of top ranking induced patterns and extracted entities are shown in Table 9. Seed list sizes and experimental results are shown in Table 3. The precision numbers shown in Table 3 were obtained by manually evaluating 100 randomly selected instances from each of the extended lists.

Category	Seed Size	Patterns Used	Extended Size	Precision
LOC	379	29	3001	70%
ORG	1597	276	33369	85%
PER	3616	265	86265	88%

Table 3: Results of LOC, ORG & PER entity list extension experiment with η_{pattern} = 10 set manually.

The overlap[4] between the induced ORG list and the Fortune-500 list has 357 organization names, which is significantly higher than the seed list overlap of 22 (see Section 1). This shows that we have been able to improve coverage considerably.

4.2 Watch Brand Name

A total of 17 watch brand names were used as seeds. In addition to the pattern scoring scheme

[3] A few locally available entities in each category were also added. These seeds are available upon request from the authors.

[4] Using same matching criteria as in Section 1.

of Section 3.1, only patterns containing sequence *"watch"* were finally retained. Entities extracted with $\eta_{\text{entity}} = 2$ are shown in Table 5. Extraction precision is 85.7%.

Corum, Longines, Lorus, Movado, Accutron, Audemars Piguet, Cartier, Chopard, Franck Muller, IWC, Jaeger-LeCoultre, A. Lange & Sohne, Patek Philippe, Rolex, Ulysse, Nardin, Vacheron Constantin

Table 4: Watch brand name seeds.

Rolex	Fossil	Swatch
Cartier	Tag Heuer	Super Bowl
Swiss	Chanel	SPOT
Movado	Tiffany	Sekonda
Seiko	TechnoMarine	Rolexes
Gucci	Franck Muller	Harry Winston
Patek Philippe	Versace	Hampton Spirit
Piaget	Raymond Weil	Girard Perregaux
Omega	Guess	Frank Mueller
Citizen	Croton	David Yurman
Armani	Audemars Piguet	Chopard
DVD	DVDs	Chinese
Breitling	Montres Rolex	Armitron
Tourneau	CD	NFL

Table 5: Extended list of watch brand names after single iteration of pattern induction algorithm.

This experiment is interesting for several reasons. First, it shows that the method presented in this paper is effective even with small number of seed instances. From this we conclude that the unambiguous nature of seed instances is much more important than the size of the seed list. Second, no negative information was used during pattern ranking in this experiment. This suggests that for relatively unambiguous categories, it is possible to successfully rank patterns using positive instances only.

4.3 Extended Lists as Features in a Tagger

Supervised models normally outperform unsupervised models in extraction tasks. The downside of supervised learning is expensive training data. On the other hand, massive amounts of unlabeled data are readily available. The goal of semi-supervised learning to combine the best of both worlds. Recent research have shown that improvements in supervised taggers are possible by including features derived from unlabeled data (Miller et al., 2004; Liang, 2005; Ando and Zhang, 2005). Similarly, automatically generated entity lists can be used as additional features in a supervised tagger.

System	F1 (Precision, Recall)
Florian et al. (2003), best single, no list	89.94 (91.37, 88.56)
Zhang and Johnson (2003), no list	90.26 (91.00, 89.53)
CRF baseline, no list	89.52 (90.39, 88.66)

Table 6: Baseline comparison on 4 categories (LOC, ORG, PER, MISC) on Test-a dataset.

For this experiment, we started with a conditional random field (CRF) (Lafferty et al., 2001) tagger with a competitive baseline (Table 6). The baseline tagger was trained[5] on the full CoNLL-2003 shared task data. We experimented with the LOC, ORG and PER lists that were automatically generated in Section 4.1. In Table 7, we show the accuracy of the tagger for the entity types for which we had induced lists. The test conditions are just baseline features with no list membership, baseline plus seed list membership features, and baseline plus induced list membership features. For completeness, we also show in Table 8 accuracy on the full CoNLL task (four entity types) without lists, with seed list only, and with the three induced lists. The seed lists (Section 4.1) were prepared from training data itself and hence with increasing training data size, the model overfitted as it became completely reliant on these seed lists. From Tables 7 & 8 we see that incorporation of token membership in the extended lists as additional membership features led to improvements across categories and at all sizes of training data. This also shows that the extended lists are of good quality, since the tagger is able to extract useful evidence from them.

Relatively small sizes of training data pose interesting learning situation and is the case with practical applications. It is encouraging to observe that the list features lead to significant improvements in such cases. Also, as can be seen from Table 7 & 8, these lists are effective even with mature taggers trained on large amounts of labeled data.

[5]Standard orthographic information, such as character n-grams, capitalization, tokens in immediate context, chunk tags, and POS were used as features.

Training Data (Tokens)	Test-a			Test-b		
	No List	Seed List	Unsup. List	No List	Seed List	Unsup. List
9268	68.16	70.91	**72.82**	60.30	63.83	**65.56**
23385	78.36	79.21	**81.36**	71.44	72.16	**75.32**
46816	82.08	80.79	**83.84**	76.44	75.36	**79.64**
92921	85.34	83.03	**87.18**	81.32	78.56	**83.05**
203621	89.71	84.50	**91.01**	84.03	78.07	**85.70**

Table 7: CRF tagger F-measure on LOC, ORG, PER extraction.

Training Data (Tokens)	Test-a			Test-b		
	No List	Seed List	Unsup. List	No List	Seed List	Unsup. List
9229	68.27	70.93	**72.26**	61.03	64.52	**65.60**
204657	89.52	84.30	**90.48**	83.17	77.20	**84.52**

Table 8: CRF tagger F-measure on LOC, ORG, PER and MISC extraction.

5 Related Work

The method presented in this paper is similar in many respects to some of the previous work on context pattern induction (Riloff and Jones, 1999; Agichtein and Gravano, 2000; Lin et al., 2003; Etzioni et al., 2005), but there are important differences. Agichtein and Gravano (2000) focus on relation extraction while we are interested in entity extraction. Moreover, Agichtein and Gravano (2000) depend on an entity tagger to initially tag unlabeled data whereas we do not have such requirement. The pattern learning methods of Riloff and Jones (1999) and the generic extraction patterns of Etzioni et al. (2005) use language-specific information (for example, chunks). In contrast, the method presented here is language independent. For instance, the English pattern induction system presented here was applied on German data without any change. Also, in the current method, induced automata compactly represent all induced patterns. The patterns induced by Riloff and Jones (1999) extract NPs and that determines the number of tokens to include in a single extraction. We avoid using such language dependent chunk information as the patterns in our case include right[6] boundary tokens thus explicitly specifying the slot in which an entity can occur. Another interesting deviation here from previous work on context pattern induction is the fact that on top of extending seed lists at high precision, we have successfully included membership in these automatically generated lexicons as features in a high quality named entity tagger improving its performance.

6 Conclusion

We have presented a novel language-independent context pattern induction method. Starting with a few seed examples, the method induces in an unsupervised way context patterns and extends the seed list by extracting more instances of the same category at fairly high precision from unlabeled data. We were able to improve a CRF-based high quality named entity tagger by using membership in these automatically generated lists as additional features.

Pattern and entity ranking methods need further investigation. Thorough comparison with previously proposed methods also needs to be carried out. Also, it will be interesting to see whether the features generated in this paper complement some of the other methods (Miller et al., 2004; Liang, 2005; Ando and Zhang, 2005) that also generate features from unlabeled data.

7 Acknowledgements

We thank the three anonymous reviewers as well as Wojciech Skut, Vrishali Wagle, Louis Monier, and Peter Norvig for valuable suggestions. This work is supported in part by NSF grant EIA-0205448.

[6] In case of predictive left context.

Induced LOC Patterns
troops in -ENT-to
Cup qualifier against -ENT-in
southern -ENT-town
war - torn -ENT-.
countries including -ENT-.
Bangladesh and -ENT-,
England in -ENT-in
west of -ENT-and
plane crashed in -ENT-.
Cup qualifier against -ENT-,

Extracted LOC Entities
US
United States
Japan
South Africa
China
Pakistan
France
Mexico
Israel
Pacific

Induced PER Patterns
compatriot -ENT-.
compatriot -ENT-in
Rep. -ENT-,
Actor -ENT-is
Sir -ENT-,
Actor -ENT-,
Tiger Woods , -ENT-and
movie starring -ENT-.
compatriot -ENT-and
movie starring -ENT-and

Extracted PER Entities
Tiger Woods
Andre Agassi
Lleyton Hewitt
Ernie Els
Serena Williams
Andy Roddick
Retief Goosen
Vijay Singh
Jennifer Capriati
Roger Federer

Induced ORG Patterns
analyst at -ENT-.
companies such as -ENT-.
analyst with -ENT-in
series against the -ENT-tonight
Today 's Schaeffer 's Option Activity Watch features -ENT-(
Cardinals and -ENT-,
sweep of the -ENT-with
joint venture with -ENT-(
rivals -ENT-Inc.
Friday night 's game against -ENT-.

Extracted ORG Entities
Boston Red Sox
St. Louis Cardinals
Chicago Cubs
Florida Marlins
Montreal Expos
San Francisco Giants
Red Sox
Cleveland Indians
Chicago White Sox
Atlanta Braves

Table 9: Top ranking LOC, PER, ORG induced pattern and extracted entity examples.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*.

Rie Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of ACL-2005*. Ann Arbor, USA.

Boris Chidlovskii. 2000. Wrapper generation by k-reversible grammar induction. *ECAI Workshop on Machine Learning for Information Extraction*.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web - an experimental study. *Artificial Intelligence Journal*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of CoNLL-2003*.

Dayne Freitag. 1997. Using grammatical inference to improve precision in information extraction. In *ICML-97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition, Nashville*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML 2001*.

Percy Liang. 2005. Semi-supervised learning for natural language. *MEng. Thesis, MIT*.

Winston Lin, Roman Yangarber, and Ralph Grishman. 2003. Bootstrapped learning of semantic classes from positive and negative examples. In *Proceedings of ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data*.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of HLT-NAACL 2004*.

L. R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *Proc. of IEEE, 77, 257–286*.

Ellen Riloff and Rosie Jones. 1999. Learning Dictionaries for Information Extraction by Multi-level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. 2005. Unsupervised learning of natural languages. In *Proceedings of National Academy of Sciences. 102:11629-11634*.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of EMNLP 2002*.

Tong Zhang and David Johnson. 2003. A robust risk minimization based named entity recognition system. In *Proceedings of CoNLL-2003*.

CoNLL-X shared task on Multilingual Dependency Parsing

Sabine Buchholz
Speech Technology Group
Cambridge Research Lab
Toshiba Research Europe
Cambridge CB2 3NH, UK
sabine.buchholz@crl.toshiba.co.uk

Erwin Marsi
Communication & Cognition
Tilburg University
5000 LE Tilburg, The Netherlands
e.c.marsi@uvt.nl

Abstract

Each year the Conference on Computational Natural Language Learning (CoNLL)[1] features a shared task, in which participants train and test their systems on exactly the same data sets, in order to better compare systems. The tenth CoNLL (CoNLL-X) saw a shared task on Multilingual Dependency Parsing. In this paper, we describe how treebanks for 13 languages were converted into the same dependency format and how parsing performance was measured. We also give an overview of the parsing approaches that participants took and the results that they achieved. Finally, we try to draw general conclusions about multi-lingual parsing: What makes a particular language, treebank or annotation scheme easier or harder to parse and which phenomena are challenging for any dependency parser?

Acknowledgement

Many thanks to Amit Dubey and Yuval Krymolowski, the other two organizers of the shared task, for discussions, converting treebanks, writing software and helping with the papers.[2]

[1] see http://ilps.science.uva.nl/~erikt/signll/conll/

[2] Thanks also to Alexander Yeh for additional help with the paper reviews. His work was made possible by the MITRE Corporation's Sponsored Research Program.

1 Introduction

Previous CoNLL shared tasks focused on NP chunking (1999), general chunking (2000), clause identification (2001), named entity recognition (2002, 2003), and semantic role labeling (2004, 2005). This shared task on full (dependency) parsing is the logical next step. Parsing is an important preprocessing step for many NLP applications and therefore of considerable practical interest. It is a complex task and as it is not straightforwardly mappable to a "classical" segmentation, classification or sequence prediction problem, it also poses theoretical challenges to machine learning researchers.

During the last decade, much research has been done on data-driven parsing and performance has increased steadily. For training these parsers, syntactically annotated corpora (treebanks) of thousands to tens of thousands of sentences are necessary; so initially, research has focused on English. During the last few years, however, treebanks for other languages have become available and some parsers have been applied to several different languages. See Section 2 for a more detailed overview of related previous research.

So far, there has not been much comparison between different dependency parsers on exactly the same data sets (other than for English). One of the reasons is the lack of a de-facto standard for an evaluation metric (labeled or unlabeled, separate root accuracy?), for splitting the data into training and testing portions and, in the case of constituency treebanks converted to dependency format, for this conversion. Another reason are the various annotation

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 149–164, New York City, June 2006.

schemes and logical data formats used by different treebanks, which make it tedious to apply a parser to many treebanks. We hope that this shared task will improve the situation by introducing a uniform approach to dependency parsing. See Section 3 for the detailed task definition and Section 4 for information about the conversion of all 13 treebanks.

In this shared task, participants had two to three months[3] to implement a parsing system that could be trained for all these languages and four days to parse unseen test data for each. 19 participant groups submitted parsed test data. Of these, all but one parsed all 12 required languages and 13 also parsed the optional Bulgarian data. A wide variety of parsing approaches were used: some are extensions of previously published approaches, others are new. See Section 5 for an overview.

Systems were scored by computing the **labeled attachment score** (LAS), i.e. the percentage of "scoring" tokens for which the system had predicted the correct head and dependency label. Punctuation tokens were excluded from scoring. Results across languages and systems varied widely from 37.8% (worst score on Turkish) to 91.7% (best score on Japanese). See Section 6 for detailed results.

However, variations are consistent enough to allow us to draw some general conclusions. Section 7 discusses the implications of the results and analyzes the remaining problems. Finally, Section 8 describes possible directions for future research.

2 Previous research

Tesnière (1959) introduced the idea of a dependency tree (a "stemma" in his terminology), in which words stand in direct head-dependent relations, for representing the syntactic structure of a sentence. Hays (1964) and Gaifman (1965) studied the formal properties of **projective** dependency grammars, i.e. those where dependency links are not allowed to cross. Mel'čuk (1988) describes a multistratal dependency grammar, i.e. one that distinguishes between several types of dependency relations (morphological, syntactic and semantic). Other theories related to dependency grammar are word grammar (Hudson, 1984) and link grammar (Sleator and Temperley, 1993).

Some relatively recent rule-based full dependency parsers are Kurohashi and Nagao (1994) for Japanese, Oflazer (1999) for Turkish, Tapanainen and Järvinen (1997) for English and Elworthy (2000) for English and Japanese.

While phrase structure parsers are usually evaluated with the GEIG/PARSEVAL measures of precision and recall over constituents (Black et al., 1991), Lin (1995) and others have argued for an alternative, dependency-based evaluation. That approach is based on a conversion from constituent structure to dependency structure by recursively defining a head for each constituent.

The same idea was used by Magerman (1995), who developed the first "head table" for the Penn Treebank (Marcus et al., 1994), and Collins (1996), whose constituent parser is internally based on probabilities of bilexical dependencies, i.e. dependencies between two words. Collins (1997)'s parser and its reimplementation and extension by Bikel (2002) have by now been applied to a variety of languages: English (Collins, 1999), Czech (Collins et al., 1999), German (Dubey and Keller, 2003), Spanish (Cowan and Collins, 2005), French (Arun and Keller, 2005), Chinese (Bikel, 2002) and, according to Dan Bikel's web page, Arabic.

Eisner (1996) introduced a data-driven dependency parser and compared several probability models on (English) Penn Treebank data. Kudo and Matsumoto (2000) describe a dependency parser for Japanese and Yamada and Matsumoto (2003) an extension for English. Nivre's parser has been tested for Swedish (Nivre et al., 2004), English (Nivre and Scholz, 2004), Czech (Nivre and Nilsson, 2005), Bulgarian (Marinov and Nivre, 2005) and Chinese Cheng et al. (2005), while McDonald's parser has been applied to English (McDonald et al., 2005a), Czech (McDonald et al., 2005b) and, very recently, Danish (McDonald and Pereira, 2006).

3 Data format, task definition

The training data derived from the original treebanks (see Section 4) and given to the shared task participants was in a simple column-based format that is

[3] Some though had significantly less time: One participant registered as late as six days before the test data release (registration was a prerequisite to obtain most of the data sets) and still went on to submit parsed test data in time.

an extension of Joakim Nivre's Malt-TAB format[4] for the shared task and was chosen for its processing simplicity. All the sentences are in one text file and they are separated by a blank line after each sentence. A sentence consists of one or more tokens. Each token is represented on one line, consisting of 10 fields. Fields are separated from each other by a TAB.[5] The 10 fields are:

1) ID: Token counter, starting at 1 for each new sentence.

2) FORM: Word form or punctuation symbol. For the Arabic data only, FORM is a concatenation of the word in Arabic script and its transliteration in Latin script, separated by an underscore. This representation is meant to suit both those that do and those that do not read Arabic.

3) LEMMA: Lemma or stem (depending on the particular treebank) of word form, or an underscore if not available. Like for the FORM, the values for Arabic are concatenations of two scripts.

4) CPOSTAG: Coarse-grained part-of-speech tag, where the tagset depends on the treebank.

5) POSTAG: Fine-grained part-of-speech tag, where the tagset depends on the treebank. It is identical to the CPOSTAG value if no POSTAG is available from the original treebank.

6) FEATS: Unordered set of syntactic and/or morphological features (depending on the particular treebank), or an underscore if not available. Set members are separated by a vertical bar (|).

7) HEAD: Head of the current token, which is either a value of ID, or zero ('0') if the token links to the virtual root node of the sentence. Note that depending on the original treebank annotation, there may be multiple tokens with a HEAD value of zero.

8) DEPREL: Dependency relation to the HEAD. The set of dependency relations depends on the particular treebank. The dependency relation of a token with HEAD=0 may be meaningful or simply 'ROOT' (also depending on the treebank).

9) PHEAD: Projective head of current token, which is either a value of ID or zero ('0'), or an underscore if not available. The dependency structure resulting from the PHEAD column is guaranteed to be projective (but is not available for all data sets), whereas the structure resulting from the HEAD column will be non-projective for some sentences of some languages (but is always available).

10) PDEPREL: Dependency relation to the PHEAD, or an underscore if not available.

As should be obvious from the description above, our format assumes that each token has exactly one head. Some dependency grammars, and also some treebanks, allow tokens to have more than one head, although often there is a distinction between primary and optional secondary relations, e.g. in the Danish Dependency Treebank (Kromann, 2003), the Dutch Alpino Treebank (van der Beek et al., 2002b) and the German TIGER treebank (Brants et al., 2002). For this shared task we decided to ignore any additional relations. However the data format could easily be extended with additional optional columns in the future. Cycles do not occur in the shared task data but are scored as normal if predicted by parsers. The character encoding of all data files is Unicode (specifically UTF-8), which is the only encoding to cover all languages and therefore ideally suited for multilingual parsing.

While the training data contained all 10 columns (although sometimes only with dummy values, i.e. underscores), the test data given to participants contained only the first 6. Participants' parsers then predicted the HEAD and DEPREL columns (any predicted PHEAD and PDEPREL columns were ignored). The predicted values were compared to the gold standard HEAD and DEPREL.[6] The official evaluation metric is the **labeled attachment score** (LAS), i.e. the percentage of "scoring" tokens for which the system has predicted the correct HEAD and DEPREL. The evaluation script defines a non-scoring token as a token where all characters of the FORM value have the Unicode category property "Punctuation".[7]

[4] http://w3.msi.vxu.se/ nivre/research/MaltXML.html

[5] Consequently, field values cannot contain TABs. In the shared task data, field values are also not supposed to contain any other whitespace (although unfortunately some spaces slipped through in the Spanish data).

[6] The official scoring script `eval.pl`, data sets for some languages and instructions on how to get the rest, the software used for the treebank conversions, much documentation, full results and other related information will be available from the permanent URL `http://depparse.uvt.nl` (also linked from the CoNLL web page).

[7] See `man perlunicode` for the technical details and the shared task website for our reasons for this decision. Note that an underscore and a percentage sign also have the Unicode "Punctuation" property.

We tried to take a test set that was representative of the genres in a treebank and did not cut through text samples. We also tried to document how we selected this set.[8] We aimed at having roughly the same size for the test sets of all languages: 5,000 scoring tokens. This is not an exact requirement as we do not want to cut sentences in half. The relatively small size of the test set means that even for the smallest treebanks the majority of tokens is available for training, and the equal size means that for the overall ranking of participants, we can simply compute the score on the concatenation of all test sets.

4 Treebanks and their conversion

In selecting the treebanks, practical considerations were the major factor. Treebanks had to be actually available, large enough, have a license that allowed free use for research or kind treebank providers who temporarily waived the fee for the shared task, and be suitable for conversion into the common format within the limited time. In addition, we aimed at a broad coverage of different language families.[9] As a general rule, we did not manually correct errors in treebanks if we discovered some during the conversion, see also Buchholz and Green (2006), although we did report them to the treebank providers and several got corrected by them.

4.1 Dependency treebanks

We used the following six dependency treebanks: **Czech**: Prague Dependency Treebank[10] (PDT) (Böhmová et al., 2003); **Arabic**: Prague Arabic Dependency Treebank[11] (PADT) (Hajič et al., 2004; Smrž et al., 2002); **Slovene**: Slovene Dependency Treebank[12] (SDT) (Džeroski et al., 2006); **Danish**: Danish Dependency Treebank[13] (Kromann, 2003); **Swedish**: Talbanken05[14] (Teleman, 1974; Einarsson, 1976; Nilsson et al., 2005); **Turkish**: Metu-Sabancı treebank[15] (Oflazer et al., 2003; Atalay et al., 2003).

The conversion of these treebanks was the easiest task as the linguistic representation was already what we needed, so the information only had to be converted from SGML or XML to the shared task format. Also, the relevant information had to be distributed appropriately over the CPOSTAG, POSTAG and FEATS columns.

For the Swedish data, no predefined distinction into coarse and fine-grained PoS was available, so the two columns contain identical values in our format. For the Czech data, we sampled both our training and test data from the official "training" partition because only that one contains gold standard PoS tags, which is also what is used in most other data sets. The Czech DEPREL values include the suffixes to mark coordination, apposition and parenthesis, while these have been ignored during the conversion of the much smaller Slovene data. For the Arabic data, sentences with missing annotation were filtered out during the conversion.

The Turkish treebank posed a special problem because it analyzes each word as a sequence of one or more inflectional groups (IGs). Each IG consists of either a stem or a derivational suffix plus all the inflectional suffixes belonging to that stem/derivational suffix. The head of a whole word is not just another word but a specific IG of another word.[16] One can easily map this representation to one in which the head of a word is a word but that

[8] See the shared task website for a more detailed discussion.

[9] That was also the reason why we decided not to include a fifth Germanic language (English) although the freely available SUSANNE treebank (Sampson, 1995) or possibly the Penn Treebank would have qualified otherwise.

[10] Many thanks to Jan Hajič for granting the temporary license for CoNLL-X and talking to LDC about it, to Christopher Cieri for arranging distribution through LDC and to Tony Castelletto for handling the distribution.

[11] Many thanks to Yuval Krymolowski for converting the treebank, Otakar Smrž for valuable help during the conversion and thanks again to Jan Hajič, Christopher Cieri and Tony Castelletto.

[12] Many thanks to the SDT people for granting the special license for CoNLL-X and to Tomaž Erjavec for converting the treebank for us.

[13] Many thanks to Matthias Trautner Kromann and assistants for creating the DDT and releasing it under the GNU General Public License and to Joakim Nivre, Johan Hall and Jens Nilsson for the conversion of DDT to Malt-XML.

[14] Many thanks to Jens Nilsson, Johan Hall and Joakim Nivre for the conversion of the original Talbanken to Talbanken05 and for making it freely available for research purposes and to Joakim Nivre again for prompt and proper respons to all our questions.

[15] Many thanks to Bilge Say and Kemal Oflazer for granting the license for CoNLL-X and answering questions and to Gülşen Eryiğit for making many corrections to the treebank and discussing some aspects of the conversion.

[16] This is a bit like saying that in "the usefulness of X for Y", "for Y" links to "use-" and not to "usefulness". Only that in Turkish, "use", "full" and "ness" each could have their own inflectional suffixes attached to them.

mapping would lose information and it is not clear whether the result is linguistically meaningful, practically useful, or even easier to parse because in the original representation, each IG has its own PoS and morphological features, so it is not clear how that information should be represented if all IGs of a word are conflated. We therefore chose to represent each IG as a separate token in our format. To make the result a connected dependency structure, we defined the HEAD of each non-word-final IG to be the following IG and the DEPREL to be "DERIV". We assigned the stem of the word to the first IG's LEMMA column, with all non-first IGs having LEMMA '_', and the actual word form to the last IG, with all non-last IGs having FORM '_'. As already mentioned in Section 3, the underscore has the punctuation character property, therefore non-last IGs (whose HEAD and DEPREL were introduced by us) are not scoring tokens. We also attached or reattached punctuation (see the README available at the shared task website for details.)

4.2 Phrase structure with functions for all constituents

We used the following five treebanks of this type: **German**: TIGER treebank[17] (Brants et al., 2002); **Japanese**: Japanese Verbmobil treebank[18] (Kawata and Bartels, 2000); **Portuguese**: The Bosque part of the Floresta sintá(c)tica[19] (Afonso et al., 2002); **Dutch**: Alpino treebank[20] (van der Beek et al., 2002b; van der Beek et al., 2002a); **Chinese**: Sinica treebank[21] (Chen et al., 2003).

Their conversion to dependency format required the definition of a head table. Fortunately, in contrast to the Penn Treebank for which the head table is based on POS[22] we could use the grammatical functions annotated in these treebanks. Therefore, head rules are often of the form: the head child of a VP/clause is the child with the HD/predicator/hd/Head function. The DEPREL value for a token is the function of the biggest constituent of which this token is the lexical head. If the constituent comprising the complete sentence did not have a function, we gave its lexical head token the DEPREL "ROOT".

For the Chinese treebank, most functions are not grammatical functions (such as "subject", "object") but semantic roles (such as "agent", "theme"). For the Portuguese treebank, the conversion was complicated by the fact that a detailed specification existed which tokens should be the head of which other tokens, e.g. the finite verb must be the head of the subject and the complementzier but the main verb must be the head of the complements and adjuncts.[23] Given that the Floresta sintá(c)tica does not use traditional VP constituents but rather verbal chunks (consisting mainly of verbs), a simple Magerman-Collins-style head table was not sufficient to derive the required dependency structure. Instead we used a head table that defined several types of heads (syntactic, semantic) and a link table that specified what linked to which type of head.[24]

Another problem existed with the Dutch treebank. Its original PoS tag set is very coarse and the PoS and the word stem information is not very reliable.[25] We therefore decided to retag the treebank automatically using the Memory-Based Tagger (MBT) (Daelemans et al., 1996) which uses a very fine-grained tag set. However, this created a problem with multiwords. MBT does not have the concept of multiwords and therefore tags all of their

[17] Many thanks to the TIGER team for allowing us to use the treebank for the shared task and to Amit Dubey for converting the treebank.

[18] Many thanks to Yasuhiro Kawata, Julia Bartels and colleagues from Tübingen University for the construction of the original Verbmobil treebank for Japanese and to Sandra Kübler for providing the data and granting the special license for CoNLL-X.

[19] Many thanks to Diana Santos, Eckhard Bick and other Floresta sint(c)tica project members for creating the treebank and making it publicly available, for answering many questions about the treebank (Diana and Eckhard), for correcting problems and making new releases (Diana), and for sharing scripts and explaining the head rules implemented in them (Eckhard). Thanks also to Jason Baldridge for useful discussions and to Ben Wing for independently reporting problems which Diana then fixed.

[20] Many thanks to Gertjan van Noord and the other people at the University of Groningen for creating the Alpino Treebank and releasing it for free, to Gertjan van Noord for answering all our questions and for providing extra test material and to Antal van den Bosch for help with the memory-based tagger.

[21] Many thanks to Academia Sinica for granting the temporary license for CoNLL-X, to Keh-Jiann Chen for answering our questions and to Amit Dubey for converting the treebank.

[22] containing rules such as: the head child of a VP is the leftmost "to", or else the leftmost past tense verb, or else etc.

[23] Eckhard Bick, p.c.

[24] See the conversion script `bosque2MALT.py` and the README file at the shared task website for details.

[25] http://www.let.rug.nl/vannoord/trees/Papers/diffs.pdf

components individually. As Alpino does not provide an internal structure for multiwords, we had to treat multiwords as one token. However, we then lack a proper PoS for the multiword. After much discussion, we decided to assign each multiword the CPOSTAG "MWU" (multiword unit) and a POSTAG which is the concatenation of the PoS of all the components as predicted by MBT (separated by an underscore). Likewise, the FEATS are a concatenation of the morphological features of all components. This approach resulted in many different POSTAG values for the training set and even in unseen values in the test set. It remains to be tested whether our approach resulted in data sets better suited for parsing than the original.

4.3 Phrase structure with some functions

We used two treebanks of this type: **Spanish**: Cast3LB[26] (Civit Torruella and Martí Antonín, 2002; Navarro et al., 2003; Civit et al., 2003); **Bulgarian**: BulTreeBank[27] (Simov et al., 2002; Simov and Osenova, 2003; Simov et al., 2004; Osenova and Simov, 2004; Simov et al., 2005).

Converting a phrase structure treebank with only a few functions to a dependency format usually requires linguistic competence in the treebank's language in order to create the head table and missing function labels. We are grateful to Chanev et al. (2006) for converting the BulTreeBank to the shared task format and to Montserrat Civit for providing us with a head table and a function mapping for Cast3LB.[28]

4.4 Data set characteristics

Table 1 shows details of all data sets. Following Nivre and Nilsson (2005) we use the following definition: "an arc (i, j) is projective iff all nodes occurring between i and j are dominated by i (where dominates is the transitive closure of the arc relation)".[29]

5 Approaches

Table 2 tries to give an overview of the wide variety of parsing approaches used by participants. We refer to the individual papers for details. There are several dimensions along which to classify approaches.

5.1 Top-down, bottom-up

Phrase structure parsers are often classified in terms of the parsing order: top-down, bottom-up or various combinations. For dependency parsing, there seem to be two different interpretations of the term "bottom-up". Nivre and Scholz (2004) uses this term with reference to Yamada and Matsumoto (2003), whose parser has to find all children of a token before it can attach that token to its head. We will refer to this as "bottom-up-trees". Another use of "bottom-up" is due to Eisner (1996), who introduced the notion of a "span". A span consists of a potential dependency arc r between two tokens i and j and all those dependency arcs that would be spanned by r, i.e. all arcs between tokens k and l with $i \leq k, l \leq j$. Parsing in this order means that the parser has to find all children and siblings on one side of a token before it can attach that token to a head on the same side. This approach assumes projective dependency structures. Eisner called this approach simply "bottom-up", while Nivre, whose parser implicitly also follows this order, called it "top-down/bottom-up" to distinguish it from the pure "bottom-up(-trees)" order of Yamada and Matsumoto (2003). To avoid confusion, we will refer to this order as "bottom-up-spans".

5.2 Unlabeled parsing versus labeling

Given that the parser needs to predict the HEAD as well as the DEPREL value, different approaches are possible: predict the (probabilities of the) HEADs of all tokens first, or predict the (probabilities of the) DEPRELs of all tokens first, or predict the HEAD and DEPREL of one token before predicting these values for the next token. Within the first approach, each dependency can be labeled independently (Corston-Oliver and Aue, 2006) or a

[26] Many thanks to Montserrat Civit and Toni Martí for allowing us to use Cast3LB for CoNLL-X and to Amit Dubey for converting the treebank.

[27] Many thanks to Kiril Simov and Petya Osenova for allowing us to use the BulTreeBank for CoNLL-X.

[28] Although unfortunately, due to a bug, the function list was not used and the Spanish data in the shared task ended up with many DEPREL values being simply '_'. By the time we discovered this, the test data release date was very close and we decided not to release new bug-fixed training material that late.

[29] Thanks to Joakim Nivre for explaining this.

	Ar	Ch	Cz	Da	Du	Ge	Ja	Po	Sl	Sp	Sw	Tu	Bu
lang. fam.	Sem.	Sin.	Sla.	Ger.	Ger.	Ger.	Jap.	Rom.	Sla.	Rom.	Ger.	Ura.	Sla.
genres	1: ne	6	3	8+	5+	1: ne	1: di	1: ne	1: no	9	4+	8	12
annotation	d	c+f	d	d	dc+f	dc+f	c+f	dc+f	d	c(+f)	dc+f/d	d	c+t
training data													
tokens (k)	54	337	1249	94	195	700	151	207	29	89	191	58	190
%non-scor.	8.8	[a]0.8	14.9	13.9	11.3	11.5	11.6	14.2	17.3	12.6	11.0	[b]33.1	14.4
units (k)	1.5	57.0	72.7	5.2	13.3	39.2	17.0	9.1	1.5	3.3	11.0	5.0	12.8
tokens/unit	[c]37.2	[d]5.9	17.2	18.2	14.6	17.8	[e]8.9	22.8	18.7	27.0	17.3	11.5	14.8
LEMMA	[f](+)	–	+	–	+	–	–	+	+	+	–	+	–
CPOSTAGs	14	13+9	12	10	13	[g]52	20	15	11	15	37	14	11
POSTAGs	19	[h]294+9	63	24	[i]302	52	77	21	28	38	37	30	53
FEATS	19	–	61	47	81	–	4	146	51	33	–	82	50
DEPRELs	27	82	78	52	26	46	7	55	25	21	56	25	18
D.s H.=0	15	1	14	1	1	1	1	6	6	1	1	1	1
%HEAD=0	5.5	16.9	6.7	6.4	8.9	6.3	18.6	5.1	5.9	4.2	6.5	13.4	7.9
%H. preced.	82.9	24.8	50.9	75.0	46.5	50.9	8.9	60.3	47.2	60.8	52.8	6.2	62.9
%H. follow.	11.6	58.2	42.4	18.6	44.6	42.7	72.5	34.6	46.9	35.1	40.7	80.4	29.2
H.=0/unit	1.9	1.0	1.0	1.0	1.2	1.0	1.5	1.0	[j]0.9	1.0	1.0	1.0	1.0
%n.p. arcs	0.4	0.0	1.9	1.0	5.4	2.3	[k]1.1	1.3	1.9	[l]0.1	1.0	1.5	0.4
%n.p. units	11.2	0.0	23.2	15.6	36.4	27.8	5.3	18.9	22.2	1.7	9.8	11.6	5.4
test data													
scor. tokens	4990	4970	5000	5010	4998	5008	5003	5009	5004	4991	5021	5021	5013
%new form	17.3	9.3	5.2	18.1	20.7	6.5	0.96	11.6	22.0	14.7	18.0	41.4	14.5
%new lem.	4.3	n/a	1.8	n/a	15.9	n/a	n/a	7.8	9.9	9.7	n/a	13.2	n/a

Table 1: Characteristics of the data sets for the 13 languages (abbreviated by their first two letters): language family (Semitic, Sino-Tibetan, Slavic, Germanic, Japonic (or language isolate), Romance, Ural-Altaic); number of genres, and genre if only one (news, dialogue, novel); type of annotation (d=dependency, c=constituents, dc=discontinuous constituents, +f=with functions, +t=with types). For the training data: number of tokens (times 1000); percentage of non-scoring tokens; number of parse tree units (usually sentences, times 1000); average number of (scoring and non-scoring) tokens per parse tree unit; whether a lemma or stem is available; how many different CPOSTAG values, POSTAG values, FEATS components and DEPREL values occur for scoring tokens; how many different values for DEPREL scoring tokens with HEAD=0 can have (if that number is 1, there is one designated label (e.g. "ROOT") for tokens with HEAD=0); percentage of scoring tokens with HEAD=0, a head that precedes or a head that follows the token (this nicely shows which languages are predominantly head-initial or head-final); the average number of scoring tokens with HEAD=0 per parse tree unit; the percentage of (scoring and non-scoring) non-projective relations and of parse tree units with at least one non-projective relation. For the test data: number of scoring tokens; percentage of scoring tokens with a FORM or a LEMMA that does not occur in the training data.

[a] final punctuation was deliberately left out during the conversion (as it is explicitly excluded from the tree structure)

[b] the non-last IGs of a word are non-scoring, see Section 4.1

[c] in many cases the parse tree unit in PADT is not a sentence but a paragraph

[d] in many cases the unit in Sinica is not a sentence but a comma-separated clause or phrase

[e] the treebank consists of transcribed dialogues, in which some sentences are very short, e.g. just "Hai." ("Yes.")

[f] only part of the Arabic data has non-underscore values for the LEMMA column

[g] no mapping from fine-grained to coarse-grained tags was available; same for Swedish

[h] 9 values are typos; POSTAGs also encode subcategorization information for verbs and some semantic information for conjunctions and nouns; some values also include parts in square brackets which in hindsight should maybe have gone to FEATS

[i] due to treatment of multiwords

[j] probably due to some sentences consisting only of non-scoring tokens, i.e. punctuation

[k] these are all disfluencies, which are attached to the virtual root node

[l] from co-indexed items in the original treebank; same for Bulgarian

	algorithm	ver.	hor.	search	lab.	non-proj	learner	pre	post	opt
all pairs										
McD	MST/Eisner	b-s	irr.	opt/approx.	2nd	+ [a]	MIRA	–	–	–
Cor	MST/Eisner	b-s	irr.	optimal	2nd	–	BPM[b]+ME [SVM]	+ [c]	–	–
Shi	MST/CLE	irr.	irr.	optimal	1st	+, CLE	MIRA	–	–	–
Can	own algorithm	irr.	irr.	approx.(?)	int.	+ [d]	TiMBL	–	–	+
Rie	ILP	irr.	irr.	increment.	int.	+ [e]	MIRA	–	–	+
Bic	CG-inspired	mpf	mpf	backtrack(?)	int.	+ [f]	MLE(?)	+ [g]	+ [h]	–
stepwise										
Dre	hag[i]/Eisner/rerank	b-s	irr.	best 1st exh	2nd	–	MLE	–	–	+ [j]
Liu	own algorithm	b-t	mpf	det./local	int.	–	MLE	–	–	–
Car	Eisner	b-s	irr.	approx.	int.	–	perceptron	–	–	–
stepwise: classifier-based										
Att	Y&M	b-t	for.	determin.	int.	+ [k]	ME [MBL,SVM,...]	stem	–	–
Cha	Y&M	b-t	for.	local	2nd	– [l]	perceptron (SNoW)	proj	–	–
Yur	own algorithm	b-s	irr.	determin.	int.	–	decision list (GPA)[m]	–	–	–
Che	chunker+Nivre	b-s	for.	determin.	int.[n]	–	SVM + ME [CRF]	–	–	–
Niv	Nivre	b-s	for.	determin.	int.	+, ps-pr	SVM	proj	deproj	+
Joh	Nivre+MST/CLE	b-s	f+b[o]	N-best	int.[p]	+, CLE	SVM (LIBSVM)	–	–	
Wu	Nivre+root parser	b-s	f/b[q]	det.[+exh.]	int.	– [+]	SVM (SVMLight)	–	[+] [r]	–
other										
Sch	PCFG/CKY	b-t	irr.	opt.	int.	+, traces	MLE [ME]	d2c	c2d	–

Table 2: Overview of parsing approaches taken by participating groups (identified by the first three letters of the first author): algorithm (Y&M: Yamada and Matsumoto (2003), ILP: Integer Linear Programming), vertical direction (irrelevant, mpf: most probable first, bottom-up-spans, bottom-up-trees), horizontal direction (irrelevant, mpf: most probable first, forward, backward), search (optimal, approximate, incremental, best-first exhaustive, deterministic), labeling (interleaved, separate and 1st step, separate and 2nd step), non-projective (ps-pr: through pseudo-projective approach), learner (ME: Maximum Entropy; learners in brackets were explored but not used in the official submission), preprocessing (projectivize, d2c: dependencies to constituents), postprocessing (deprojectivize, c2d: constituents to dependencies), learner parameter optimization per language

[a] non-projectivity through approximate search, used for some languages
[b] 20 averaged perceptrons combined into a Bayes Point Machine
[c] introduced a single POS tag "aux" for all Swedish auxiliary and model verbs
[d] by having no projectivity constraint
[e] selective projectivity constraint for Japanese
[f] several approaches to non-projectivity
[g] using some FEATS components to create some finer-grained POSTAG values
[h] reattachment rules for some types of non-projectivity
[i] head automaton grammar
[j] determined the maximally allowed distance for relations
[k] through special parser actions
[l] pseudo-projectivizing training data only
[m] Greedy Prepend Algorithm
[n] but two separate learners used for unlabeled parsing versus labeling
[o] both foward and backward, then combined into a single tree with CLE
[p] but two separate SVMs used for unlabeled parsing versus labeling
[q] forward parsing for Japanese and Turkish, backward for the rest
[r] attaching remaining unattached tokens through exhaustive search (not for submitted runs)

sequence classifier can label all children of a token together (McDonald et al., 2006). Within the third approach, HEAD and DEPREL can be predicted simultaneously, or in two separate steps (potentially using two different learners).

5.3 All pairs

At the highest level of abstraction, there are two fundamental approaches, which we will call "all pairs" and "stepwise". In an "all pairs" approach, every possible pair of two tokens in a sentence is considered and some score is assigned to the possibility of this pair having a (directed) dependency relation. Using that information as building blocks, the parser then searches for the best parse for the sentence. This approach is one of those described in Eisner (1996). The definition of "best" parse depends on the precise model used. That model can be one that defines the score of a complete dependency tree as the sum of the scores of all dependency arcs in it. The search for the best parse can then be formalized as the search for the maximum spanning tree (MST) (McDonald et al., 2005b). If the parse has to be projective, Eisner's bottom-up-span algorithm (Eisner, 1996) can be used for the search. For non-projective parses, McDonald et al. (2005b) propose using the Chu-Liu-Edmonds (CLE) algorithm (Chu and Liu, 1965; Edmonds, 1967) and McDonald and Pereira (2006) describe an approximate extension of Eisner's algorithm. There are also alternatives to MST which allow imposing additional constraints on the dependency structure, e.g. that at most one dependent of a token can have a certain label, such as "subject", see Riedel et al. (2006) and Bick (2006). By contrast, Canisius et al. (2006) do not even enforce the tree constraint, i.e. they allow cycles. In a variant of the "all pairs" approach, only those pairs of tokens are considered that are not too distant (Canisius et al., 2006).

5.4 Stepwise

In a stepwise approach, not all pairs are considered. Instead, the dependency tree is built stepwise and the decision about what step to take next (e.g. which dependency to insert) can be based on information about, in theory all, previous steps and their results (in the context of generative probabilistic parsing, Black et al. (1993) call this the history). Stepwise approaches can use an explicit probability model over next steps, e.g. a generative one (Eisner, 1996; Dreyer et al., 2006), or train a machine learner to predict those. The approach can be deterministic (at each point, one step is chosen) or employ various types of search. In addition, parsing can be done in a bottom-up-constituent or a bottom-up-spans fashion (or in another way, although this was not done in this shared task). Finally, parsing can start at the first or the last token of a sentence. When talking about languages that are written from left to right, this distinction is normally referred to as left-to-right versus right-to-left. However, for multilingual parsing which includes languages that are written from right to left (Arabic) or sometimes top to bottom (Chinese, Japanese) this terminology is confusing because it is not always clear whether a left-to-right parser for Arabic would really start with the leftmost (i.e. last) token of a sentence or, like for other languages, with the first (i.e. rightmost). In general, starting with the first token ("forward") makes more sense from a psycholinguistic point of view but starting with the last ("backward") might be beneficial for some languages (possibly related to them being head-initial versus head-final languages). The parsing order directly determines what information will be available from the history when the next decision needs to be made. Stepwise parsers tend to interleave the prediction of HEAD and DEPREL.

5.5 Non-projectivity

All data sets except the Chinese one contain some non-projective dependency arcs, although their proportion varies from 0.1% to 5.4%. Participants took the following approaches to non-projectivity:

- Ignore, i.e. predict only projective parses. Depending on the way the parser is trained, it might be necessary to at least projectivize the training data (Chang et al., 2006).
- Always allow non-projective arcs, by not imposing any projectivity constraint (Shimizu, 2006; Canisius et al., 2006).
- Allow during parsing under certain conditions, e.g. for tokens with certain properties (Riedel et al., 2006; Bick, 2006) or if no alternative projective arc has a score above the threshold

(Bick, 2006) or if the classifier chooses a special action (Attardi, 2006) or the parser predicts a trace (Schiehlen and Spranger, 2006).

- Introduce through post-processing, e.g. through reattachment rules (Bick, 2006) or if the change increases overall parse tree probability (McDonald et al., 2006).

- The pseudo-projective approach (Nivre and Nilsson, 2005): Transform non-projective training trees to projective ones but encode the information necessary to make the inverse transformation in the DEPREL, so that this inverse transformation can also be carried out on the test trees (Nivre et al., 2006).

5.6 Data columns used

Table 3 shows which column values have been used by participants. Nobody used the PHEAD/PDEPREL column in any way. It is likely that those who did not use any of the other columns did so mainly for practical reasons, such as the limited time and/or the difficulty to integrate it into an existing parser.

5.6.1 FORM versus LEMMA

Lemma or stem information has often been ignored in previous dependency parsers. In the shared task data, it was available in just over half the data sets. Both LEMMA and FORM encode lexical information. There is therefore a certain redundancy. Participants have used these two columns in different ways:

- Use only one (see Table 3).

- Use both, in different features. Typically, a feature selection routine and/or the learner itself (through weights) will decide about the importance of the resulting features.

- Use a variant of the FORM as a substitute for a missing LEMMA. Bick (2006) used the lowercased FORM if the LEMMA is not available, Corston-Oliver and Aue (2006) a prefix and Attardi (2006) a stem derived by a rule-based system for Danish, German and Swedish.

	form	lem.	cpos	pos	feats
McD	++ [a]	+ [b]	–?	+	+, co+cr.pr.
Cor	+	+	+ [c]	++	+, co+cr.pr.[d]
Shi	+	–	+	–	–
Can	+	–	–	+	–
Rie	+ [e]	+	+	+ [f]	+ cr.pr.
Bic	(+)	+	+ [g]	+	(+)
Dre	++ [h]	+	rer.	rer.	–
Liu	(+)	+	++	+	–
Car	++	+	++	+	+ comp.
Att	(+)	+	+	–	(+)
Cha	–	+	–	+	+ atomic
Yur	+	+	+	+	+ comp.
Che	+	+	+	+	+ atomic?
Niv	+	+	+	+	+ comp.
Joh	+	–	+	+	+ comp.
Wu	+	–	+	+	–
Sch	?	(+)[i]	?	(+)	(+)

Table 3: Overview of data columns used by participating groups. '–': a column value was not used at all. '+': used in at least some features. '(+)': Variant of FORM used only if LEMMA is missing, or only parts of FEATS used. '++': used more extensively than another column containing related information (where FORM and LEMMA are related, as are CPOSTAG and POSTAG), e.g. also in combination features or features for context tokens in addition to features for the focus token(s). "rer.": used in the reranker only. For the last column: atomic, comp. = components, cr.pr. = cross-product.

[a] also prefix and suffix for labeler
[b] instead of form for Arabic and Spanish
[c] instead of POSTAG for Dutch and Turkish
[d] for labeler; unlab. parsing: only some for global features
[e] also prefix
[f] also 1st character of POSTAG
[g] only as backoff
[h] reranker: also suffix; if no lemma, use prefix of FORM
[i] LEMMA, POSTAG, FEATS only for back-off smoothing

5.6.2 CPOSTAG versus POSTAG

All data sets except German and Swedish had different values for CPOSTAG and POSTAG, although the granularity varied widely. Again, there are different approaches to dealing with the redundancy:

- Use only one for all languages.

- Use both, in different features. Typically, a feature selection routine and/or the learner itself (through weights) will decide about the importance of the resulting features.
- Use one or the other for each language.

5.6.3 Using FEATS

By design, a FEATS column value has internal structure. Splitting it at the '|'[30] results in a set of components. The following approaches have been used:

- Ignore the FEATS.
- Treat the complete FEATS value as atomic, i.e. do not split it into components.
- Use only some components, e.g. Bick (2006) uses only case, mood and pronoun subclass and Attardi (2006) uses only gender, number, person and case.
- Use one binary feature for each component. This is likely to be useful if grammatical function is indicated by case.
- Use one binary feature for each cross-product of the FEATS components of i and the FEATS components of j. This is likely to be useful for agreement phenomena.
- Use one binary feature for each FEATS component of i that also exists for j. This is a more explicit way to model agreement.

5.7 Types of features

When deciding whether there should be a dependency relation between tokens i and j, all parsers use at least information about these two tokens. In addition, the following sources of information can be used (see Table 4): token context (**tc**): a limited number (determined by the window size) of tokens directly preceding or following i or j; **ch**ildren: information about the already found children of i and j; **si**blings: in a set-up where the decision is not "is there a relation between i and j" but "is i the head of j" or in a separate labeling step, the siblings of i are the already found children of j; structural context (**sc**) other than children/siblings: neighboring subtrees/spans, or ancestors of i and j; **di**stance from i to j; information derived from all the tokens **in** between i and j (e.g. whether there is an intervening verb or how many intervening commas there are); **gl**obal features (e.g. does the sentence contain a finite verb); explicit feature **co**mbinations (depending on the learner, these might not be necessary, e.g. a polynomial kernel routinely combines features); for classifier-based parsers: the previous **ac**tions, i.e. classifications; whether information about **la**bels is used as input for other decisions. Finally, the precise set of features can be **op**timized per language.

[30] or for Dutch, also at the '_'

	tc	ch	si	sc	di	in	gl	co	ac	la	op
McD	+	l	+	l	?	l	l	+	–	l	(+)[a]
Cor	+	l[b]	l	+	p	–	+	+	–	–	(+)[c]
Shi	+	–	–	–	+	–	–	+	–	+	–
Can	+	–	–	–	+	–	–	–	–	–	–
Rie	+	–	+[d]	–	?	?	–	+	–	+[e]	+
Bic	+	+[f]	+[g]	–	+	+[h]	–	+	–	++	(+)[i]
Dre	r	r	+	r	+	r	–	+	–	r	r
Liu	–	+	–	+	+	–	–	+	–	–	–
Car	+	–	+	–	+	+	–	+	–	+	–
Att	–	+	+	+	–	–	–	–	+	+	(+)[j]
Cha	+	+	–	l	–	–	–	+	+	–	–
Yur	+	+	–	?	–	–	–	–	–	–	+
Che	–	+	+	+	+	–	–	–	–	–	–
Niv	+	+	–	+	–	–	–	–	–	+	+
Joh	+	+	–	+	–	–	–	–	–	+	–
Wu	+	+	–	+	–	–	–	+	–	+	–
Sch	–	+	–	–	–	–	–	–	–	+	–

Table 4: Overview of features used by participating groups. See the text for the meaning of the column abbreviations. For separate HEAD and DEPREL assignment: p: only for unlabeled parsing, l: only for labeling, r: only for reranking.

[a] FORM versus LEMMA
[b] number of tokens governed by child
[c] POSTAG versus CPOSTAG
[d] for arity constraint
[e] for arity constraint
[f] for "full" head constraint
[g] for uniqueness constraint
[h] for barrier constraint
[i] of constraints
[j] POS window size

6 Results

Table 5 shows the official results for submitted parser outputs.[31] The two participant groups with the highest total score are McDonald et al. (2006) and Nivre et al. (2006). As both groups had much prior experience in multilingual dependency parsing (see Section 2), it is not too surprising that they both achieved good results. It is surprising, however, how similar their total scores are, given that their approaches are quite different (see Table 2). The results show that experiments on just one or two languages certainly give an indication of the usefulness of a parsing approach but should not be taken as proof that one algorithm is better for "parsing" (in general) than another that performs slightly worse. The Bulgarian scores suggest that rankings would not have been very different had it been the 13th obligatory languages.

Table 6 shows that the same holds had we used another evaluation metric. Note that a negative number in both the third and fifth column indicates that errors on HEAD and DEPREL occur together on the same token more often than for other parsers. Finally, we checked that, had we also scored on punctuation tokens, total scores as well as rankings would only have shown very minor differences.

7 Result analysis

7.1 Across data sets

The average LAS over all data sets varies between 56.0 for Turkish and 85.9 for Japanese. Top scores vary between 65.7 for Turkish and 91.7 for Japanese. In general, there is a high correlation between the best scores and the average scores. This means that data sets are inherently easy or difficult, no matter what the parsing approach. The "easiest" one is clearly the Japanese data set. However, it would be wrong to conclude from this that Japanese in general is easy to parse. It is more likely that the effect stems from the characteristics of the data. The Japanese Verbmobil treebank contains dialogue within a restricted domain (making business appointments). As can be seen in Table 1, there are very few new FORM values in the test data, which is an indication of many dialogues in the treebank being similar. In addition, parsing units are short on average. Finally, the set of DEPREL values is very small and consequently the ratio between (C)POSTAG and DEPREL values is extremely favorable. It would be interesting to apply the shared task parsers to the Kyoto University Corpus (Kurohashi and Nagao, 1997), which is the standard treebank for Japanese and has also been used by Kudo and Matsumoto

[31] Unfortunately, urgent other obligations prevented two participants (John O'Neil and Kenji Sagae) from submitting a paper about their shared task work. Their results are indicated by a smaller font. Sagae used a best-first probabilistic version of Y&M (p.c.).

	LAS	unlabeled		label acc.	
McD	80.3	=	86.6	−1	86.7
Niv	80.2	=	85.5	+1	86.8
O'N	78.4	=	85.3	−1	85.0
Rie	77.9	=	85.0	−1	84.9
Sag	77.8	−2	83.7	+2	85.6
Che	77.7	+1	84.6	=	84.2
Cor	76.9	+1	84.4	−1	84.0
Cha	76.8	=	83.5	+1	84.1
Joh	74.9	−1	80.4	=	83.7
Car	74.7	+1	81.2	=	83.5
Wu	71.7	−1	78.4	−1	79.1
Can	70.8	+1	78.4	−1	78.6
Bic	70.0	=	77.5	[a]+2	80.3
Dre	65.2	−1	74.5	−1	75.2
Yur	65.0	−1	73.5	−2	70.9
Liu	63.3	−2	70.7	=	73.6
Sch	62.8	=	72.1	[b]+3	75.7
Att	61.2	[c]+4	76.2	=	70.7
Shi	34.2	=	38.7	=	39.7

Table 6: Differences in ranking depending on the evaluation metric. The second column repeats the official metric (LAS). The third column shows how the ranking for each participant changes (or not: '=') if the unlabeled attachment scores, as shown in the fourth column, are used. The fifth column shows how the ranking changes (in comparison to LAS) if the label accuracies, as shown in the sixth column, are used.

[a] In Bick's method, preference is given to the assignment of dependency labels.

[b] Schiehlen derived the constituent labels for his PCFG approach from the DEPREL values.

[c] Due to the bug (see footnote with Table 5).

	Ar	Ch	Cz	Da	Du	Ge	Ja	Po	Sl	Sp	Sw	Tu	Tot	SD	Bu
McD	**66.9**	85.9	**80.2**	**84.8**	**79.2**	**87.3**	90.7	**86.8**	**73.4**	**82.3**	82.6	63.2	**80.3**	8.4	**87.6**
Niv	**66.7**	86.9	78.4	**84.8**	**78.6**	85.8	**91.7**	**87.6**	70.3	**81.3**	**84.6**	**65.7**	**80.2**	8.5	**87.4**
O'N	**66.7**	86.7	76.6	82.8	77.5	85.4	90.6	84.7	71.1	79.8	81.8	57.5	78.4	9.4	85.2
Rie	**66.7**	**90.0**	67.4	83.6	**78.6**	86.2	90.5	84.4	71.2	77.4	80.7	58.6	77.9	10.1	0.0
Sag	62.7	84.7	75.2	81.6	76.6	84.9	90.4	**86.0**	69.1	77.7	82.0	63.2	77.8	9.0	0.0
Che	65.2	84.3	76.2	81.7	71.8	84.1	89.9	85.1	71.4	80.5	81.1	61.2	77.7	8.7	86.3
Cor	63.5	79.9	74.5	81.7	71.4	83.5	90.0	84.6	**72.4**	80.4	79.7	61.7	76.9	8.5	83.4
Cha	60.9	85.1	72.9	80.6	72.9	84.2	89.1	84.0	69.5	79.7	82.3	60.5	76.8	9.4	0.0
Joh	64.3	72.5	71.5	81.5	72.7	80.4	85.6	84.6	66.4	78.2	78.1	63.4	74.9	7.7	0.0
Car	60.9	83.7	68.8	79.7	67.3	82.4	88.1	83.4	68.4	77.2	78.7	58.1	74.7	9.7	83.3
Wu	63.8	74.8	59.4	78.4	68.5	76.5	90.1	81.5	67.8	73.0	71.7	55.1	71.7	9.7	79.7
Can	57.6	78.4	60.9	77.9	74.6	77.6	87.4	77.4	59.2	68.3	79.2	51.1	70.8	11.1	78.7
Bic	55.4	76.2	63.0	74.6	69.5	74.7	84.8	78.2	64.3	71.4	74.1	53.9	70.0	9.3	79.2
Dre	53.4	71.6	60.5	66.6	61.6	71.0	82.9	75.3	58.7	67.6	67.6	46.1	65.2	9.9	74.8
Yur	52.4	72.7	51.9	71.6	62.8	63.8	84.4	70.4	55.1	69.6	65.2	60.3	65.0	9.5	73.5
Liu	50.7	75.3	58.5	77.7	59.4	68.1	70.8	71.1	57.2	65.1	63.8	41.7	63.3	10.4	67.6
Sch	44.4	66.2	53.3	76.1	72.1	68.7	83.4	71.0	50.7	47.0	71.1	49.8	62.8	13.0	0.0
Att	53.8	54.9	59.8	66.4	58.2	69.8	65.4	75.4	57.2	67.4	68.8	37.8	[a]61.2	9.9	72.9
Shi	62.8	0.0	0.0	75.8	0.0	0.0	0.0	0.0	64.6	73.2	79.5	54.2	34.2	36.3	0.0
Av	59.9	78.3	67.2	78.3	70.7	78.6	85.9	80.6	65.2	73.5	76.4	56.0			80.0
SD	6.5	8.8	8.9	5.5	6.7	7.5	7.1	5.8	6.8	8.4	6.5	7.7			6.3

Table 5: Labeled attachment scores of parsers on the 13 test sets. The total score (Tot) and standard deviations (SD) from the average per participant are calculated over the 12 obligatory languages (i.e. excluding Bulgarian). Note that due to the equal sizes of the test sets for all languages, the total scores, i.e. the LAS over the concatenation of the 12 obligatory test sets, are identical (up to the first decimal digit) to the average LAS over the 12 test sets. Averages and standard deviations per data set are calculated ignoring zero scores (i.e. results not submitted). The highest score for each column and those not significantly worse ($p < 0.05$) are shown in bold face. Significance was computed using the official scoring script `eval.pl` and Dan Bikel's Randomized Parsing Evaluation Comparator, which implements stratified shuffling.

[a] Attardi's submitted results contained an unfortunate bug which caused the DEPREL values of all tokens with HEAD=0 to be an underscore (which is scored as incorrect). Using the simple heuristic of assigning the DEPREL value that most frequently occured with HEAD=0 in training would have resulted in a total LAS of 67.5.

(2000), or to the domain-restricted Japanese dialogues of the ATR corpus (Lepage et al., 1998).[32]

Other relatively "easy" data sets are Portuguese (2nd highest average score but, interestingly, the third-longest parsing units), Bulgarian (3rd), German (4th) and Chinese (5th). Chinese also has the second highest top score[33] and Chinese parsing units are the shortest. and Chinese parsing units are the shortest. We note that all "easier" data sets offer large to middle-sized training sets.

The most difficult data set is clearly the Turkish one. It is rather small, and in contrast to Arabic and Slovene, which are equally small or smaller, it covers 8 genres, which results in a high percentage of new FORM and LEMMA values in the test set. It is also possible that parsers get confused by the high proportion (one third!) of non-scoring tokens

[32] Unfortunately, both these treebanks need to be bought, so they could not be used for the shared task. Note also that Japanese dependency parsers often operate on "bunsetsus" instead of words. Bunsetsus are related to chunks and consist of a content word and following particles (if any).

[33] Although this seems to be somewhat of a mystery compared to the ranking according to the average scores. Riedel et al. (2006)'s top score is more than 3% absolute above the second highest score and they offer no clear explanation for their success.

and the many tokens with '_' as either the FORM or LEMMA. There is a clear need for further research to check whether other representations result in better performance.

The second-most difficult data set is Arabic. It is quite small and has by far the longest parsing units. The third-most difficult data set is Slovene. It has the smallest training set. However, its average as well as top score far exceed those for Arabic and Turkish, which are larger. Interestingly, although the treebank text comes from a single source (a translation of Orwell's novel "1984"), there is quite a high proportion of new FORM and LEMMA values in the test set. The fourth-most difficult data set is Czech in terms of the average score and Dutch in terms of the top score. The diffence in ranking for Czech is probably due to the fact that it has by far the largest training set and ironically, several participants could not train on all data within the limited time, or else had to partition the data and train one model for each partition. Likely problems with the Dutch data set are: noisy (C)POSTAG and LEMMA, (C)POSTAG for multiwords, and the highest proportion of non-projectivity.

Factors that have been discussed so far are: the size of the training data, the proportion of new FORM and LEMMA values in the test set, the ratio of (C)POSTAG to DEPREL values, the average length of the parsing unit the proportion of non-projective arcs/parsing units. It would be interesting to derive a formula based on those factors that fits the shared task data and see how well it predicts results on new data sets. One factor that seems to be irrelevant is the head-final versus head-initial distinction, as both the "easiest" and the most difficult data sets are for head-final languages. There is also no clear proof that some language families are easier (with current parsing methods) than others. It would be interesting to test parsers on the Hebrew treebank (Sima'an et al., 2001), to compare performance to Arabic, the other Semitic language in the shared task, or on the Hungarian Szeged Corpus (Csendes et al., 2004), for another agglutinative language.

7.2 Across participants

For most parsers, their ranking for a specific language differs at most a few places from their overall ranking. There are some outliers though. For example, Johansson and Nugues (2006) and Yuret (2006) are seven ranks higher for Turkish than overall, while Riedel et al. (2006) are five ranks lower. Canisius et al. (2006) are six and Schiehlen and Spranger (2006) even eight ranks higher for Dutch than overall, while Riedel et al. (2006) are six ranks lower for Czech and Johansson and Nugues (2006) also six for Chinese. Some of the higher rankings could be related to native speaker competence and resulting better parameter tuning but other outliers remain a mystery. Even though McDonald et al. (2006) and Nivre et al. (2006) obtained very similar overall scores, a more detailed look at their performance shows clear differences. Taken over all 12 obligatory languages, both obtain a recall of more than 89% on root tokens (i.e. those with HEAD=0) but Nivre's precision on them is much lower than McDonald's (80.91 versus 91.07). This is likely to be an effect of the different parsing approaches.

7.3 Across part-of-speech tags

When breaking down by part-of-speech the results of all participants on all data sets, one can observe some patterns of "easy" and "difficult" parts-of-speech, at least in so far as tag sets are comparable across treebanks. The one PoS that everybody got 100% correct are the German infinitival markers (tag PTKZU; like "to" in English). Accuracy on the Swedish equivalent (IM) is not far off at 98%. Other easy PoS are articles, with accuracies in the nineties for German, Dutch, Swedish, Portuguese and Spanish. As several participants have remarked in their papers, prepositions are much more difficult, with typical accuracies in the fifties or sixties. Similarly, conjunctions typically score low, with accuracies even in the forties for Arabic and Dutch.

8 Future research

There are many directions for interesting research building on the work done in this shared task. One is the question which factors make data sets "easy" or difficult. Another is finding out how much of parsing performance depends on annotations such as the lemma and morphological features, which are not yet routinely part of treebanking efforts. In this respect, it would be interesting to repeat ex-

periments with the recently released new version of the TIGER treebank which now contains this information. One line of research that does not require additional annotation effort is defining or improving the mapping from coarse-grained to fine-grained PoS tags.[34] Another is harvesting and using large-scale distributional data from the internet. We also hope that by combining parsers we can achieve even better performance, which in turn would facilitate the semi-automatic enlargement of existing treebanks and possibly the detection of remaining errors. This would create a positive feedback loop. Finally one must not forget that almost all of the LEMMA, (C)POSTAG and FEATS values and even part of the FORM column (the multiword tokens used in many data sets and basically all tokenization for Chinese and Japanese, where words are normally not delimited by spaces) have been manually created or corrected and that the general parsing task has to integrate automatic tokenization, morphological analysis and tagging. We hope that the resources created and lessons learned during this shared task will be valuable for many years to come but also that they will be extended and improved by others in the future, and that the shared task website will grow into an informational hub on multilingual dependency parsing.

References

A. Arun and F. Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: The case of French. In *Proc. of the 43rd Annual Meeting of the ACL*.

D. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proc. of the Human Language Technology Conf. (HLT)*.

E. Black, S. Abney, D. Flickenger, et al. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California*.

E. Black, F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, and S. Roukos. 1993. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proc. of the 31rd Annual Meeting of the ACL*.

S. Buchholz and D. Green. 2006. Quality control of treebanks: documenting, converting, patching. In *LREC 2006 workshop on Quality assurance and quality measurement for language and speech resources.*

[34]For the Swedish Talbanken05 corpus, that work has been done after the shared task (see the treebank's web site).

A. Chanev, K. Simov, P. Osenova, and S. Marinov. 2006. Dependency conversion and parsing of the BulTreeBank. In *Proc. of the LREC-Workshop Merging and Layering Linguistic Information*.

Y. Cheng, M. Asahara, and Y. Matsumoto. 2005. Chinese deterministic dependency analyzer: Examining effects of global features and root node finder. In *Proc. of SIGHAN-2005*.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

M. Collins, J. Hajic, L. Ramshaw, and C. Tillmann. 1999. A statistical parser for Czech. In *Proc. of the 37th Annual Meeting of the ACL*.

M. Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proc. of the 34th Annual Meeting of the ACL*.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of the 35th Annual Meeting of the ACL*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

B. Cowan and M. Collins. 2005. Morphology and reranking for the statistical parsing of Spanish. In *Proc. of the Joint Conf. on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

D. Csendes, J. Csirik, and T. Gyimóthy. 2004. The Szeged corpus: a POS tagged and syntactically annotated Hungarian natural language corpus. In *Proc. of the 5th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*.

W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A memory-based part of speech tagger-generator. In *Proc. of the 4th Workshop on Very Large Corpora (VLC)*.

A. Dubey and F. Keller. 2003. Probabilistic parsing for German using sister-head dependencies. In *Proc. of the 41st Annual Meeting of the ACL*.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

J. Einarsson. 1976. Talbankens skriftspråkskonkordans.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of the 16th Intern. Conf. on Computational Linguistics (COLING)*, pages 340–345.

D. Elworthy. 2000. A finite-state parser with dependency structure output. In *Proc. of the 6th Intern. Workshop on Parsing Technologies (IWPT)*.

H. Gaifman. 1965. Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337.

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

R. Hudson. 1984. *Word Grammar*. Blackwell.

T. Kudo and Y. Matsumoto. 2000. Japanese dependency structure analysis based on support vector machines. In *Proc. of the Joint Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*.

S. Kurohashi and M. Nagao. 1994. KN parser: Japanese dependency/case structure analyzer. In *Proceedings of the Workshop on Sharable Natural Language*, pages 48–55.

S. Kurohashi and M. Nagao. 1997. Kyoto University text corpus project. In *Proc. of the 5th Conf. on Applied Natural Language Processing (ANLP)*, pages 115–118.

Y. Lepage, S. Ando, S. Akamine, and H. Iida. 1998. An annotated corpus in Japanese using Tesnière's structural syntax. In *ACL-COLING Workshop on Processing of Dependency-Based Grammars*, pages 109–115.

D. Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*.

D. Magerman. 1995. Statistical decision-tree models for parsing. In *Proc. of the 33rd Annual Meeting of the ACL*, pages 276–283.

M. Marcus, G. Kim, M. Marcinkiewicz, R. Mac-Intyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *Proc. of the Workshop on Human Language Technology (HLT)*.

S. Marinov and J. Nivre. 2005. A data-driven dependency parser for Bulgarian. In *Proc. of the 4th Workshop on Treebanks and Linguistic Theories (TLT)*, pages 89–100.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of the 11th Conf. of the European Chapter of the ACL (EACL)*.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. of the 43rd Annual Meeting of the ACL*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of the Joint Conf. on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

I. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. The SUNY Press, Albany, N.Y.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of the 43rd Annual Meeting of the ACL*, pages 99–106.

J. Nivre and M. Scholz. 2004. Deterministic dependency parsing of English text. In *Proc. of the 20th Intern. Conf. on Computational Linguistics (COLING)*, pages 64–70.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proc. of the 8th Conf. on Computational Natural Language Learning (CoNLL)*, pages 49–56.

K. Oflazer. 1999. Dependency parsing with an extended finite state approach. In *Proc. of the 37th Annual Meeting of the ACL*, pages 254–260.

G. Sampson. 1995. *English for the Computer: The SUSANNE Corpus and analytic scheme*. Clarendon Press.

K. Sima'an, A. Itai, Y. Winter, A. Altman, and N. Nativ. 2001. Building a tree-bank of modern Hebrew text. In *Journal Traitement Automatique des Langues (t.a.l.) — Special Issue on Natural Language Processing and Corpus Linguistics*.

D. Sleator and D. Temperley. 1993. Parsing English with a link grammar. In *Proc. of the 3rd Intern. Workshop on Parsing Technologies (IWPT)*.

P. Tapanainen and T. Järvinen. 1997. A non-projective dependency parser. In *Proc. of the 5th Conf. on Applied Natural Language Processing (ANLP)*.

U. Teleman, 1974. *Manual för grammatisk beskrivning av talad och skriven svenska (MAMBA)*.

L. Tesnière. 1959. *Elément de syntaxe structurale*. Klincksieck, Paris.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of the 8th Intern. Workshop on Parsing Technologies (IWPT)*, pages 195–206.

Papers by participants of CoNLL-X (this volume)

G. Attardi. 2006. Experiments with a multilanguage non-projective dependency parser.

E. Bick. 2006. LingPars, a linguistically inspired, language-independent machine learner for dependency treebanks.

S. Canisius, T. Bogers, A. van den Bosch, J. Geertzen, and E. Tjong Kim Sang. 2006. Dependency parsing by inference over high-recall dependency predictions.

M. Chang, Q. Do, and D. Roth. 2006. A pipeline model for bottom-up dependency parsing.

S. Corston-Oliver and A. Aue. 2006. Dependency parsing with reference to Slovene, Spanish and Swedish.

M. Dreyer, D. Smith, and N. Smith. 2006. Vine parsing and minimum risk reranking for speed and precision.

R. Johansson and P. Nugues. 2006. Investigating multilingual dependency parsing.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines.

S. Riedel, R. Çakıcı, and I. Meza-Ruiz. 2006. Multi-lingual dependency parsing with incremental integer linear programming.

M. Schiehlen and K. Spranger. 2006. Language independent probabilistic context-free parsing bolstered by machine learning.

N. Shimizu. 2006. Maximum spanning tree algorithm for non-projective labeled dependency parsing.

D. Yuret. 2006. Dependency parsing as a classication problem.

The Treebanks Used in the Shared Task

This page contains references to all the treebanks used in the CoNLL-X shared task. This page should be consulted whenever a shared task paper refers to a treebank without including the actual reference.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*. Kluwer Academic Publishers, Dordrecht.

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. "Floresta sintá(c)tica": a treebank for Portuguese. In *Proc. of the Third Intern. Conf. on Language Resources and Evaluation (LREC)*, pages 1698–1703.

N. B. Atalay, K. Oflazer, and B. Say. 2003. The annotation process in the Turkish treebank. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proc. of the First Workshop on Treebanks and Linguistic Theories (TLT)*.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

M. Civit, M^a A. Martí, B. Navarro, N. Bufi, B. Fernández, and R. Marcos. 2003. Issues in the syntactic annotation of Cast3LB. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*.

M. Civit Torruella and M^a A. Martí Antonín. 2002. Design principles for a Spanish treebank. In *Proc. of the First Workshop on Treebanks and Linguistic Theories (TLT)*.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proc. of the Fifth Intern. Conf. on Language Resources and Evaluation (LREC)*.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

Y. Kawata and J. Bartels. 2000. Stylebook for the Japanese treebank in VERBMOBIL. Verbmobil-Report 240, Seminar für Sprachwissenschaft, Universität Tübingen.

M. T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In *Proc. of the Second Workshop on Treebanks and Linguistic Theories (TLT)*.

B. Navarro, M. Civit, M^a A. Martí, R. Marcos, and B. Fernández. 2003. Syntactic, semantic and pragmatic annotation in Cast3LB. In *Proc. of the Workshop on Shallow Processing of Large Corpora (SProLaC)*.

J. Nilsson, J. Hall, and J. Nivre. 2005. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. In *Proc. of the NODALIDA Special Session on Treebanks*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15.

P. Osenova and K. Simov. 2004. BTB-TR05: BulTreeBank stylebook. BulTreeBank version 1.0. Bultreebank project technical report. Available at: http://www.bultreebank.org/TechRep/BTB-TR05.pdf.

K. Simov and P. Osenova. 2003. Practical annotation scheme for an HPSG treebank of Bulgarian. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*, pages 17–24.

K. Simov, G. Popova, and P. Osenova. 2002. HPSG-based syntactic treebank of Bulgarian (BulTreeBank). In A. Wilson, P. Rayson, and T. McEnery, editors, *A Rainbow of Corpora: Corpus Linguistics and the Languages of the World*, pages 135–142. Lincom-Europa, Munich.

K. Simov, P. Osenova, and M. Slavcheva. 2004. BTB-TR03: BulTreeBank morphosyntactic tagset. BTB-TS version 2.0. Bultreebank project technical report. Available at: http://www.bultreebank.org/TechRep/BTB-TR03.pdf.

K. Simov, P. Osenova, A. Simov, and M. Kouylekov. 2005. Design and implementation of the Bulgarian HPSG-based treebank. In *Journal of Research on Language and Computation – Special Issue*, pages 495–522. Kluwer Academic Publishers.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
page 165, New York City, June 2006.

Experiments with a Multilanguage Non-Projective Dependency Parser

Giuseppe Attardi
Dipartimento di Informatica
largo B. Pontecorvo, 3
I-56127 Pisa, Italy
attardi@di.unipi.it

1 Introduction

Parsing natural language is an essential step in several applications that involve document analysis, e.g. knowledge extraction, question answering, summarization, filtering. The best performing systems at the TREC Question Answering track employ parsing for analyzing sentences in order to identify the query focus, to extract relations and to disambiguate meanings of words.

These are often demanding applications, which need to handle large collections and to provide results in a fraction of a second. Dependency parsers are promising for these applications since a dependency tree provides predicate-argument relations which are convenient for use in the later stages. Recently statistical dependency parsing techniques have been proposed which are deterministic and/or linear (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004). These parsers are based on learning the correct sequence of Shift/Reduce actions used to construct the dependency tree. Learning is based on techniques like SVM (Vapnik 1998) or Memory Based Learning (Daelemans 2003), which provide high accuracy but are often computationally expensive. Kudo and Matsumoto (2002) report a two week learning time on a Japanese corpus of about 8000 sentences with SVM. Using Maximum Entropy (Berger, et al. 1996) classifiers I built a parser that achieves a throughput of over 200 sentences per second, with a small loss in accuracy of about 2-3 %.

The efficiency of Maximum Entropy classifiers seems to leave a large margin that can be exploited to regain accuracy by other means. I performed a series of experiments to determine whether increasing the number of features or combining several classifiers could allow regaining the best accuracy. An experiment cycle in our setting requires less than 15 minutes for a treebank of moderate size like the Portuguese treebank (Afonso et al., 2002) and this allows evaluating the effectiveness of adding/removing features that hopefully might apply also when using other learning techniques.

I extended the Yamada-Matsumoto parser to handle labeled dependencies: I tried two approaches: using a single classifier to predict pairs of actions and labels and using two separate classifiers, one for actions and one for labels.

Finally, I extended the repertoire of actions used by the parser, in order to handle non-projective relations. Tests on the PDT (Böhmovà et al., 2003) show that the added actions are sufficient to handle all cases of non-projectivity. However, since the cases of non-projectivity are quite rare in the corpus, the general learner is not supplied enough of them to learn how to classify them accurately, hence it may be worthwhile to exploit a second classifier trained specifically in handling non-projective situations.

1. Summary of the approach

The overall parsing algorithm is an inductive statistical parser, which extends the approach by Yamada and Matsumoto (2003), by adding six new reduce actions for handling non-projective relations and also performs dependency labeling.

Parsing is deterministic and proceeds bottom-up. Labeling is integrated within a single processing step.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 166–170, New York City, June 2006.

The parser is modular: it can use several learning algorithms: Maximum Entropy, SVM, Winnow, Voted Perceptron, Memory Based Learning, as well as combinations thereof. The submitted runs used Maximum Entropy and I present accuracy and performance comparisons with other learning algorithms.

No additional resources are used.

No pre-processing or post-processing is used, except stemming for Danish, German and Swedish.

2 Features

Columns from input data were used as follows.

LEMMA was used in features whenever available, otherwise the FORM was used. For Danish, German and Swedish the Snowball stemmer (Porter 2001) was used to generate a value for LEMMA. This use of stemming slightly improved both accuracy and performance.

Only CPOSTAG were used. PHEAD/PDEPREL were not used.

FEATS were used to extract a single token combining gender, number, person and case, through a language specific algorithm.

The selection of features to be used in the parser is controlled by a number of parameters. For example, the parameter `PosFeatures` determines for which tokens the POS tag will be included in the context, `PosLeftChildren` determines how many left outermost children of a token to consider, `PastActions` tells how many previous actions to include as features.

The settings used in the submitted runs are listed below and configure the parser for not using any word forms. Positive numbers refer to input tokens, negative ones to token on the stack.

```
LemmaFeatures          -2 -1 0 1 2 3
PosFeatures            -2 -1 0 1 2 3
MorphoFeatures         -1 0 1 2
DepFeatures            -1 0
PosLeftChildren        2
PosLeftChild           -1 0
DepLeftChild           -1 0
PosRightChildren       2
PosRightChild          -1 0
DepRightChild          -1
PastActions            1
```

The context for POS tags consisted of 1 token left and 3 tokens to the right of the focus words, except for Czech and Chinese were 2 tokens to the left and 4 tokens to the right were used. These values were chosen by performing experiments on the training data, using 10% of the sentences as held-out data for development.

3 Inductive Deterministic Parsing

The parser constructs dependency trees employing a deterministic bottom-up algorithm which performs Shift/Reduce actions while analyzing input sentences in left-to-right order.

Using a notation similar to (Nivre and Scholz, 2003), the state of the parser is represented by a quadruple $\langle S, I, T, A\rangle$, where S is the stack, I is the list of (remaining) input tokens, T is a stack of temporary tokens and A is the arc relation for the dependency graph.

Given an input string W, the parser is initialized to $\langle (), W, (), ()\rangle$, and terminates when it reaches a configuration $\langle S, (), (), A\rangle$.

The parser by Yamada and Matsumoto (2003) used the following actions:

`Shift` in a configuration $\langle S, n|I, T, A\rangle$, pushes n to the stack, producing the configuration $\langle n|S, I, T, A\rangle$.

`Right`[1] in a configuration $\langle s_1|S, n|I, T, A\rangle$, adds an arc from s_1 to n and pops s_1 from the stack, producing the configuration $\langle S, n|I, T, A\cup\{(s_1, r, n)\}\rangle$.

`Left` in a configuration $\langle s_1|S, n|I, T, A\rangle$, adds an arc from n to s_1, pops n from input, pops s_1 from the stack and moves it back to I, producing the configuration $\langle S, s_1|I, T, A\cup\{(n, r, s_1)\}\rangle$.

At each step the parser uses classifiers trained on treebank data in order to predict which action to perform and which dependency label to assign given the current configuration.

4 Non-Projective Relations

For handling non-projective relations, Nivre and Nilsson (2005) suggested applying a pre-processing step to a dependency parser, which consists in lifting non-projective arcs to their head repeatedly, until the tree becomes pseudo-projective. A post-processing step is then required to restore the arcs to the proper heads.

[1] Nivre and Scholz reverse the direction, while I follow here the terminology in Yamada and Matsumoto (2003).

I adopted a novel approach, which consists in adding six new parsing actions:

`Right2` in a configuration $\langle s_1|s_2|S, n|I, T, A\rangle$, adds an arc from s_2 to n and removes s_2 from the stack, producing the configuration $\langle s_1|S, n|I, T, A\cup\{(s_2, r, n)\}\rangle$.

`Left2` in a configuration $\langle s_1|s_2|S, n|I, T, A\rangle$, adds an arc from n to s_2, pops n from input, pops s_1 from the stack and moves it back to I, producing the configuration $\langle s_2|S, s_1|I, T, A\cup\{(n, r, s_2)\}\rangle$.

`Right3` in a configuration $\langle s_1|s_2|s_3|S, n|I, T, A\rangle$, adds an arc from s_3 to n and removes s_3 from the stack, producing the configuration $\langle s_1|s_2|S, n|I, T, A\cup\{(s_3, r, n)\}\rangle$.

`Left3` in a configuration $\langle s_1|s_2|s_3|S, n|I, T, A\rangle$, adds an arc from n to s_3, pops n from input, pops s_1 from the stack and moves it back to I, producing the configuration $\langle s_2|s_3|S, s_1|I, T, A\cup\{(n, r, s_3)\}\rangle$.

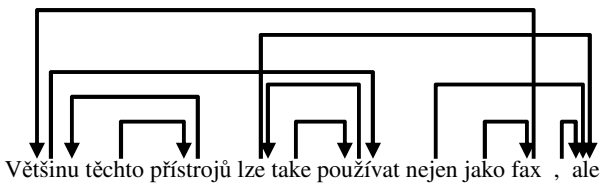
`Extract` in a configuration $\langle s_1|s_2|S, n|I, T, A\rangle$, move s_2 from the stack to the temporary stack, then `Shift`, producing the configuration $\langle n|s_1|S, I, s_2|T, A\rangle$.

`Insert` in a configuration $\langle S, I, s_1|T, A\rangle$, pops s_1 from T and pushes it to the stack, producing the configuration $\langle s_1|S, I, T, A\rangle$.

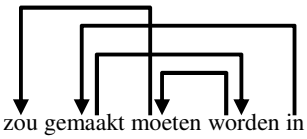
The actions `Right2` and `Left2` are sufficient to handle almost all cases of non-projectivity: for instance the training data for Czech contain 28081 non-projective relations, of which 26346 can be handled by `Left2`/`Right2`, 1683 by `Left3`/`Right3` and just 52 require `Extract`/`Insert`.

Here is an example of non-projectivity that can be handled with `Right2` (*nejen* → *ale*) and `Left3` (*fax* → *Většinu*):

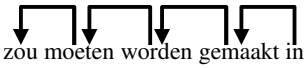
Většinu těchto přístrojů lze take používat nejen jako fax, ale současně …



The remaining cases are handled with the last two actions: `Extract` is used to postpone the creation of a link, by saving the token in a temporary stack; `Insert` restores the token from the temporary stack and resumes normal processing.



This fragment in Dutch is dealt by performing an `Extract` in configuration $\langle$*moeten*|*gemaakt*|*zou*, *worden*|*in*, $A\rangle$ followed immediately by an `Insert`, leading to the following configuration, which can be handled by normal `Shift`/`Reduce` actions:



Another linguistic phenomenon is the anticipation of pronouns, like in this Portuguese fragment:

`Tudo é possivel encontrar em o IX Salão de Antiguidades, desde objectos de ouro e prata, moedas, ...`

The problem here is due to the pronoun `Tudo` (*Anything*), which is the object of `encontrar` (*find*), but which is also the head of `desde` (*from*) and its preceding comma. In order to be able to properly link `desde` to `Tudo`, it is necessary to postpone its processing; hence it is saved with `Extract` to the temporary stack and put back later in front of the comma with `Insert`. In fact the pair `Extract`/`Insert` behaves like a generalized `Rightn`/`Leftn`, when n is not known. As in the example, except for the case where n=2, it is difficult to predict the value of n, since there can be an arbitrary long sequence of tokens before reaching the position where the link can be inserted.

5 Performance

I used my own C++ implementation of Maximum Entropy, which is very fast both in learning and classification. On a 2.8 MHz Pentium Xeon PC, the learning time is about 15 minutes for Portuguese and 4 hours for Czech. Parsing is also very fast, with an average throughput of 200 sentences per second: Table 1 reports parse time for parsing each whole test set. Using Memory Based Learning increases considerably the parsing time, while as expected learning time is quite shorter. On the other hand MBL achieves an improvement up to 5% in accuracy, as shown in detail in Table 1.

Language	Maximum Entropy						MBL				
	LAS %	Corrected LAS	UAS %	LA %	Train time sec	Parse time sec	LAS %	UAS %	LA %	Train time sec	Parse time sec
Arabic	*53.81*	54.15	*69.50*	72.97	181	2.6	**59.70**	**74.69**	75.49	24	950
Bulgarian	*72.89*	72.90	*85.24*	77.68	452	1.5	**79.17**	**85.92**	83.22	88	353
Chinese	*54.89*	70.00	*81.33*	58.75	1156	1.8	**72.17**	**83.08**	75.55	540	478
Czech	*59.76*	62.10	*73.44*	69.84	13800	12.8	**69.20**	**80.22**	77.72	496	13500
Danish	*66.35*	71.72	*78.84*	74.65	386	3.2	**76.13**	**83.65**	82.06	52	627
Dutch	*58.24*	63.71	*68.93*	66.47	679	3.3	**68.97**	**74.73**	75.93	132	923
German	*69.77*	75.88	*80.25*	78.39	9315	4.3	**79.79**	**84.31**	86.88	1399	3756
Japanese	*65.38*	78.01	*82.05*	73.68	129	0.8	**83.39**	**86.73**	89.95	44	97
Portuguese	*75.36*	79.40	*85.03*	80.79	1044	4.9	**80.97**	**86.78**	85.27	160	670
Slovene	*57.19*	60.63	*72.14*	69.36	98	3.0	**62.67**	**76.60**	72.72	16	547
Spanish	*67.44*	70.33	*74.25*	82.19	204	2.4	**74.37**	**79.70**	85.23	54	769
Swedish	*68.77*	**75.20**	*83.03*	72.42	1424	2.9	74.85	**83.73**	77.81	96	1177
Turkish	*37.80*	**48.83**	*65.25*	49.81	177	2.3	47.58	**65.25**	59.65	43	727

Table 1. Results for the CoNLL-X Shared task (official values in italics).

For details on the CoNLL-X shared task and the measurements see (Buchholz, et al. 2006).

6 Experiments

I performed several experiments to tune the parser.

I also tried alternative machine learning algorithms, including SVM, Winnow, Voted Perceptron.

The use of SVM turned out quite impractical since the technique does not scale to the size of training data involved: training an SVM with such a large number of features was impossible for any of the larger corpora. For smaller ones, e.g. Portuguese, training required over 4 days but produced a bad model which could not be used (I tried both the TinySVM (Kudo 2002) and the LIBSVM (Chang and Lin 2001) implementations).

Given the speed of the Maximum Entropy classifier, I explored whether increasing the number of features could improve accuracy. I experimented adding various features controlled by the parameters above: none appeared to be effective, except the addition of the previous action.

The classifier returns both the action and the label to be assigned. Some experiments were carried out splitting the task among several specialized classifiers. I experimented with:

1. three classifiers: one to decide between *Shift*/*Reduce*, one to decide which *Reduce* action and a third one to choose the dependency in case of *Left*/*Right* action
2. two classifiers: one to decide which action to perform and a second one to choose the dependency in case of *Left*/*Right* action

None of these variants produced improvements in precision. Only a small improvement in labeled attachment score was noticed using the full, non-specialized classifier to decide the action but discarding its suggestion for label and using a specialized classifier for labeling. However this was combined with a slight decrease in unlabeled attachment score, hence it was not considered worth the effort.

7 Error Analysis

The parser does not attempt to assign a dependency relation to the root. A simple correction of assigning a default value for each language gave an improvement in the LAS as shown in Table 1.

7.1 Portuguese

Out of the 45 dependency relations that the parser had to assign to a sentence, the largest number of

errors occurred assigning N<PRED (62), ACC (46), PIV (43), CJT (40), N< (34), P< (30).

The highest number of head error occurred at the CPOS tags PRP with 193 and V with 176. In particular just four prepositions (em, de, a, para) accounted for 120 head errors.

Most of the errors occur near punctuations. Often this is due to the fact that commas introduce relative phrases or parenthetical phrases (e.g. "o suspeito, de 38 anos, que trabalha"), that produce diversions in the flow. Since the parser makes decisions analyzing only a window of tokens of a limited size, it gets confused in creating attachments. I tried to add some global context features, to be able to distinguish these cases, in particular, a count of the number of punctuation marks seen so far, whether punctuation is present between the focus words. None of them helped improving precision and were not used in the submitted runs.

7.2 Czech

Most current parsers for Czech do not perform well on Apos (apposition), Coord (coordination) and ExD (ellipses), but they are not very frequent. The largest number of errors occur on Obj (166), Adv (155), Sb (113), Atr (98). There is also often confusion among these: 33 times Obj instead of Adv, 32 Sb instead of Obj, 28 Atr instead of Adv.

The high error rate of J (adjective) is expected, mainly due to coordination problems. The error of R (preposition) is also relatively high. Prepositions are problematic, but their error rate is higher than expected since they are, in terms of surface order, rather regular and close to the noun. It could be that the decision by the PDT to hang them as heads instead of children, causes a problem in attaching them. It seems that a post-processing may correct a significant portion of these errors.

The labels ending with _Co, _Ap or _Pa are nodes who are members of the Coordination, Apposition or the Parenthetical relation, so it may be worth while omitting these suffixes in learning and restore them by post-processing.

An experiment using as training corpus a subset consisting of just sentences which include non-projective relations achieved a LAS of 65.28 % and UAS of 76.20 %, using MBL.

Acknowledgments. Kiril Ribarov provided insightful comments on the results for Czech.

The following treebanks were used for training the parser: (Afonso et al., 2002; Atalay et al., 2003; Böhmovà et al., 2003; Brants et al., 2002; Chen et al., 2003; Civit Torruella and Martì Antonìn, 2002; Džeroski et al., 2006; Hajiç et al., 2004; Kawata and Bartels, 2000; Kromann, 2003; Nilsson et al., 2005; Oflazer et al., 2003; Simov et al., 2005; van der Beek et al., 2002).

References

A. Berger, S. Della Pietra, and M. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1).

S. Buchholz, et al. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proc. of the Tenth CoNLL.*

C.-C. Chang, C.-J. Lin. 2001. LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm/

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2003. Timbl: Tilburg memory based learner, version 5.0, reference guide. Technical Report ILK 03-10, Tilburg University, ILK.

T. Kudo. 2002. tinySVM. http://www.chasen.org/~taku/software/TinySVM/

T. Kudo, Y. Matsumoto. 2002. Japanese Dependency Analysis using Cascaded Chunking. In *Proc. of the Sixth CoNLL.*

R. McDonald, et al. 2005. Non-projective Dependency Parsing using Spanning Tree Algorithms. In *Proc. of HLT-EMNLP*.

J. Nivre, et al. 2004. Memory-based Dependency Parsing. In *Proc.s of the Eighth CoNLL*, ed. H. T. Ng and E. Riloff, Boston, Massachusetts, pp. 49–56.

J. Nivre and M. Scholz. 2004. Deterministic Dependency Parsing of English Text. In *Proc. of COLING 2004*, Geneva, Switzerland, pp. 64–70.

J. Nivre and J. Nilsson, 2005. Pseudo-Projective Dependency Parsing. In *Proc. of the 43rd Annual Meeting of the ACL*, pp. 99-106.

M.F. Porter. 2001. Snowball Stemmer. http://www.snowball.tartarus.org/

V. N. Vapnik. 1998. The Statistical Learning Theory. Springer.

H. Yamada and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proc. of the 8th International Workshop on Parsing Technologies* (*IWPT*), pp. 195–206.

LingPars, a Linguistically Inspired, Language-Independent Machine Learner for Dependency Treebanks

Eckhard Bick

Institute of Language and Communication
University of Southern Denmark
5230 Odense M, Denmark
eckhard.bick@mail.dk

Abstract

This paper presents a Constraint Grammar-inspired machine learner and parser, LingPars, that assigns dependencies to morphologically annotated treebanks in a function-centred way. The system not only bases attachment probabilities for PoS, case, mood, lemma on those features' function probabilities, but also uses topological features like function/PoS n-grams, barrier tags and daughter-sequences. In the CoNLL shared task, performance was below average on attachment scores, but a relatively higher score for function tags/deprels in isolation suggests that the system's strengths were not fully exploited in the current architecture.

1 Introduction

This paper describes LingPars, a Constraint Grammar-inspired language-independent treebank-learner developed from scratch between January 9th and March 9th 2006 in the context of the CoNLL-X 2006 shared task (http://nextens.uvt.nl/~conll/), organized by Sabine Buchholz, Erwin Marsi, Yval Krymolowski and Amit Dubey. Training treebanks and test data were provided for 13 different languages: Arabic (Smrž et al. 2002), Chinese (Chen et al. 2003), Czech (Hajič et al. 2001), Danish (Kromann 2003), Dutch (van der Beek et al. 2002), German (Brants et.al 2002), Japanese (Kawata and Bartels), Portuguese (Afonso et al. 2002), Slovene (Džerosky et al. 2006), Spanish (Palomar et al. 2004), Swedish (Nilsson et al. 2005), Turkish (Oflazer et al. 2003 and Nart et.al 2003), Bulgarian (Simov et al. 2005). A number of these treebanks were not originally annotated in dependency style, but transformed from constituent tree style for the task, and all differ widely in terms of tag granularity (21-302 part-of-speech tags, 7-82 function labels). Also, not all treebanks included morphological information, and only half offered a lemma field. Such descriptive variation proved to be a considerable constraint for our parser design, as will be explained in chapter 2. No external resources and no structural preprocessing were used[1].

2 Language independence versus theory independence

While manual annotation and/or linguistic, rule-based parsers are necessary for the creation of its training data, only a machine learning based parser (as targeted in the CoNNL shared task) can hope to be truly language independent in its design. The question is, however, if this necessarily implies independence of linguistic/descriptive theory.

In our own approach, LingPars, we thus departed from the Constraint Grammar descriptive model (Karlsson et al. 2005), where syntactic function tags (called DEPREL or dependency relations in the shared task) rank higher than dependency/constituency and are established *before* head attachments, rather than vice versa (as would be the case for many probabilistic, chunker based systems, or

[1]The only exception is what we consider a problem in the dependency-version of the German TIGER treebank, where postnominal attributes of nouns appear as dependents of that noun's head if the latter is a preposition, but not otherwise (e.g. if the head's head is a preposition). LingPars failed to learn this somewhat idiosyncratic distinction, but performance improved when the analysis was pre-processed with an additional np-layer (to be re-flattened after parsing.).

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 171–175, New York City, June 2006.

the classical PENN treebank descriptive model). In our hand-written, rule based parsers, dependency treebanks are constructed by using sequential attachment rules, generally attaching functions (e.g. subject, object, postnominal) to forms (finite verb, noun) or lexical tags (tense, auxiliary, transitive), with a direction condition and the possibility of added target, context or barrier conditions (Bick 2005).

In LingPars, we tried to mimic this methodology by trying to learn probabilities for both CG style syntactic-function contexts and function-to-form attachment rules. We could not, however, implement the straightforward idea of learning probabilities and optimal ordering for an existing body of (manual) seeding rules, because the 13 treebanks were not harmonized in their tag sets and descriptive conventions[2].

As an example, imagine a linguistic rule that triggers "subclause-hood" for a verb-headed dependency-node as soon as a subordinator attaches to it, and then, implementing "subclause-hood", tries to attach the verb not to the root, but to another verb left of the subordinator, or right to a root-attaching verb. For the given set of treebanks probabilities and ordering priorities for this rule cannot be learned by one and the same parser, simply because some treebanks attach the verb to the subordinator rather than vice versa, and for verb chains, there is no descriptive consensus as to whether the auxiliary/construction verb (e.g. Spanish) or the main verb (e.g. Swedish) is regarded as head.

3 System architecture

The point of departure for pattern learning in LingPars were the fine-grained part of speech (PoS) tags (POSTAG) and the LEMMA tag. For those languages that did not provide a lemma tag, lowercased word form was used instead. Also, where available from the FEATS field and not already integrated into the PoS tag, the following information was integrated into the PoS tag:

a) *case*, which was regarded as a good predictor for function, as well as a good dependency-indicator for e.g. preposition- and adnominal attachment

b) *mood/finiteness,* in order to predict subordination and verb chaining, especially in the absence of auxiliary class information in the FEATS field

c) *pronoun subclass,* in order to predict adnominal vs. independent function as well as subordinating function (relatives and interrogatives)

A few treebanks did not classify subordinating words as conjunctions, relatives, interrogatives etc., but lumped them into the general adverb and pronoun classes. Danish is a case in point - here, the treebank classified all non-inflecting words as PoS 'U'[3]. Our solution, implemented only for Danish and Swedish, was to introduce a list of structure-words, that would get their PoS appended with an '-S', enabling the learner to distinguish between e.g. "ordinary" ADV, and "structural" ADV-S.

3.1 The parser

In a first round, our parser calculates a preference list of functions and dependencies for each word, examining all possible mother-daughter pairs and n-grams in the sentence (or paragraph). Next, dependencies are adjusted for function, basically summing up the frequency-, distance- and direction-calibrated function→PoS attachment probabilities for all contextually allowed functions for a given word. Finally, dependency probabilities are weighted using linked probabilities for possible mother-, daughter- and sister-tags in a second pass.

The result are 2 arrays, one for possible daughter→mother pairs, one for word:function pairs. Values in both arrays are normalized to the 0..1 interval, meaning that for instance even an originally low probability, long distance attachment will get high values after normalization if there are *few or no* competing alternatives for the word in question.

LingPars then attempts to "effectuate" the dependency (daughter→mother) array, starting with the - in normalized terms - highest value[4]. If the daughter candidate is as yet unattached, and the dependency does not produce circularities or crossing branches, the corresponding part of the (ordered) word:function array is calibrated for the suggested dependency, and the top-ranking function chosen.

In principle, one pass through the dependency array would suffice to parse a sentence. However,

[2] Neither was there time (and for some languages: reading knowledge) to write the necessary converters to and from a normalized standard formalism for each treebank.

[3] For the treebank as such, no information is lost, since it will be recoverable from the function tag. In a training situation, however, there is much less to train on than in a treebank with a more syntactic definition of PoS.

[4] Though we prefer to think of attachments as bottom-up choices, the value-ordered approach is essentially neither bottom-up nor top-down, depending on the language and the salience of relations in a sentence, all runs had a great variation in the order of attachments. A middle-level attachment like case-based preposition-attachment, for instance, can easily outperform (low) article- or (high) top-node-attachment.

due to linguistic constraints like uniqueness principle, barrier tags and "full" heads[5], some words may be left unattached or create conflicts for their heads. In these cases, weights are reduced for the conflicting functions, and increased for all daughter→mother values of the unattached word. The value arrays are then recomputed and rerun. In the case of unattached words, a complete rerun is performed, allowing problematic words to attach before those words that would otherwise have blocked them. In the case of a function (e.g subject uniqueness) conflict, only the words involved in the conflict are rerun. If no conflict-free solution is found after 19 runs, barrier-, uniqueness- and projectivity-constraints are relaxed for a last run[6].

Finally, the daughter-sequence for each head (with the head itself inserted) is checked against the probability of its function sequence (learned not from n-grams proper, but from daughter-sequences in the training corpus). For instance, the constituents of a clause would make up such a sequence and allow to correct a sequence like SUBJ VFIN ARG2 ARG1 into SUBJ VFIN ARG1 ARG2, where ARG1 and ARG2 are object functions with a preferred order (for the language learned) of ARG1 ARG2.

3.2 Learning functions (deprels)

LingPars computes function probabilities (Vf, function value) at three levels: First, each lemma and PoS is assigned *local* (context-free) probabilities for all possible functions. Second, the probability of a given function occurring at a specific place in a function n-gram (func-gram, example (a)) is calculated (with n between 2 and 6). The learner only used *endocentric* func-grams, marking which of the function positions had their head within the func-gram. If no funcgram supported a given function, its probability for the word in question was set to zero. At the third level, for each endocentric n-gram of word classes (PoS), the probability for a given function occurring at a given position in the n-gram (position 2 in example (b)) was computed. Here, only the longest possible n-grams were used by the parser, and first and last positions of the n-gram were used only to provide context, not to assign function probabilities.

(a)>N→2 SUBJ→4 <N→2 AUX MV→4 ACC→5
(b) art→2 n:SUBJ→4 adj→2 v-fin v-inf→4 n→5

3.3 Learning dependencies

In a rule based Constraint Grammar system, dependency would be expressed as attachment of functions to forms (i.e. subject to verb, or modifier to adjective). However, with empty deprel fields, LingPars cannot use functions directly, only their probabilities. Therefore, in a first pass, it computes the probability for the whole possible attachment matrix for a sentence, using learned mother- and daughter-normalized frequencies for attachments of type (a) PoS→PoS, (b) PoS→Lex, (c) Lex→PoS and (d) Lex→Lex, taking into account also the learned directional and distance preferences. Each matrix cell is then filled with a value Vfa ("function attachment value") - the sum of the individual normalized probabilities of all possible functions for that particular daughter given that particular mother multiplied with the preestablished, attachment-independent Vf value for that token-function combination.

Inspired by the BARRIER conditions in CG rule contexts, our learner also records the frequency of those PoS and those functions (deprels) that may appear between a dependent of PoS A and a head of PoS B. The parser then regards all *other,* non-registered interfering PoS or functions as blocking tokens for a given attachment pair, reducing its attachment value by a factor of 1/100.

In a second pass, the attachment matrix is calibrated using the relative probabilities for dependent daughters, dependent sisters and head mother given. This way, probabilities of object and object complement sisters will enhance each other, and given the fact that treebanks differ as to which element of a verb chain arguments attach to, a verbal head can be treated differently depending on whether it has a high probability for another verb (with auxiliary, modal or main verb function) as mother or daughter or not.

Finally, like for functions, n-grams are used to calculate attachment probabilities. For each endocentric PoS n-gram (of length 6 or less), the probabilities of all treebank-supported PoS:function chains and their dependency arcs are learned, and the value for an attachment word pair occurring in the chain will be corrected using both the chain/n-gram probability and the Vf value for the function

[5]Head types with a limited maximum number of dependents (usually, one)
[6]In the rare case of still missing heads or functions, these are computed using probabilities for a simplified set of word classes (mostly the CPOSTAG), or - as a last resort - set to ROOT-attachment.

associated with the dependent in that particular chain. For contextual reasons, arcs central to the n-gram are weighted higher than peripheral arcs.[7]

3.4 Non-projectivity and other language-specific problems

As a general rule, non-projective arcs were only allowed if no other, projective head could be found for a given word. However, linguistic knowledge suggests that non-projective arcs should be particularly likely in connection with verb-chain-dependencies, where subjects attach to the finite verb, but objects to the non-finite verb, which can create crossing arcs in the case of object fronting, chain inversion etc. Since we also noted an error-risk from arguments getting attached to the closest verb in a chain rather than the linguistically correct one[8], we chose to introduce systematic, after-parse raising of certain pre-defined arguments from the auxiliary to the main verb. This feature needs language-dependent parameters, and time constraints only allowed the implementation for Danish, Spanish, Portuguese and Czech. For Dutch, we also discovered word-class-related projectivity-errors, that could be remedied by exempting certain FEATS classes from the parser's general projectivity constraint altogether (*prep-voor* and *V-hulp*)[9].

In order to improve root accuracy, topnode probability was set to zero for verbs with a safe subordinator dependent. However, even those treebanks descriptively supporting this did not all PoS-mark subordinators. Therefore, FEATS-information was used, or as a last resort - for Danish and Swedish - word forms.

A third language-specific error-source was punctuation, because some treebanks (cz, sl, es) allowed punctuation as heads. Also, experiments for the Germanic and Romance languages showed that performance decreased when punctuation was allowed as BARRIER, but increased, when a fine-grained punctuation PoS[10] was included in function and dependency n-grams.

[7]Due to BARRIER constraints, or simply because of insufficient training data in the face of a very detailed tag set, it may be impossible to assign all words n-gram supported functions or dependencies. In the former case, local function probabilities are used, in the latter attachment is computed as function → PoS probability only, using the most likely function.

[8] Single verbs being more frequent than verb chains, the learner tended to generalize close attachment, and even (grand)daughter and (grand)mother conditions could not entirely remedy this problem.

[9]Though desirable, there was no time to implement this for other languages.

[10] Only for Spanish and Swedish was there a subdivision of punctuation PoS, so we had to supply this information in all other cases by adding token-information to the POSTAG field.

4 Evaluation

Because of LingPars' strong focus on function tags, a separate analysis of attachment versus label performance was thought to be of interest. Ill. 1 plots the latter (Y-axis) against the former (X-axis), with dot size symbolizing treebank size. In this evaluation, a fixed training chunk size of 50,000 tokens[11] was used, and tested on a different sample of 5,000 tokens (see also 5/50 evaluation in ill. 2). For most languages, function performance was better than attachment performance (3.2 percentage points on average, as opposed to 0.44 for the CoNLL systems overall), with dots above the hyphenated "diagonal of balance". Interestingly, the graphics also makes it clear that performance was lower for small treebanks, despite the fact that training corpus size had been limited in the experiment, possibly indicating correlated differences in the balance between tag set size and treebank size.

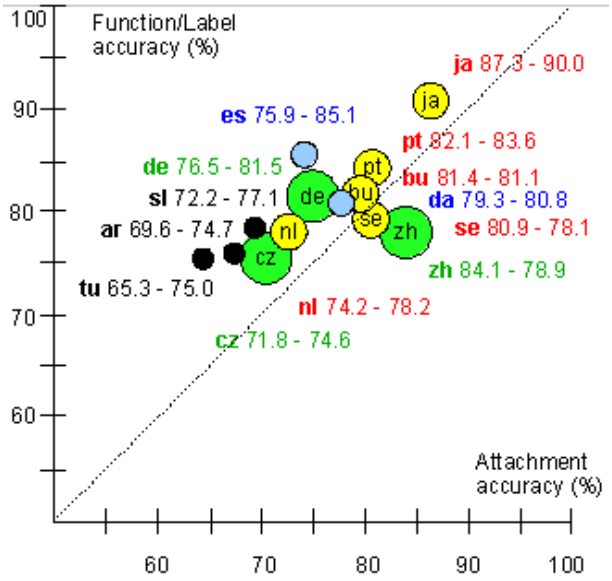


Illustration 1: Attachment accuracy (x-axis) vs. label accuracy (y-axis)

Ill. 2 keeps the information from ill. 1 (5/50-dep and 5/50-func), represented in the two lower lines, but adds performance for maximal training corpus size[12] with (a) a randomly chosen test chunk of 5,000 tokens *not included in the training corpus* (5/all-5) and (b) a 20,000 token chunk *from the training corpus* (20/all). Languages were sorted ac-

[11]Smaller for Slovene and Arabic (for these languages: largest possible)

[12]Due to deadline time constraints, an upper limit of 400,000 lines was forced on the biggest treebanks, when training for unknown test data, meaning that only ½ of the German data and 1/3 of the Czech data could be used.

cording to 20/all-func accuracy. As can be seen from the dips in the remaining (lower) curves, small training corpora (asterisk-marked languages) made it difficult for the parser (1) to match 20/all attachment performance on unknown data, and (2) to learn labels/functions in general (dips in all function curves, even 20/all). For the larger treebanks, the parser performed better (1-3 percentage points) for the full training set than for the 50,000 token training set.

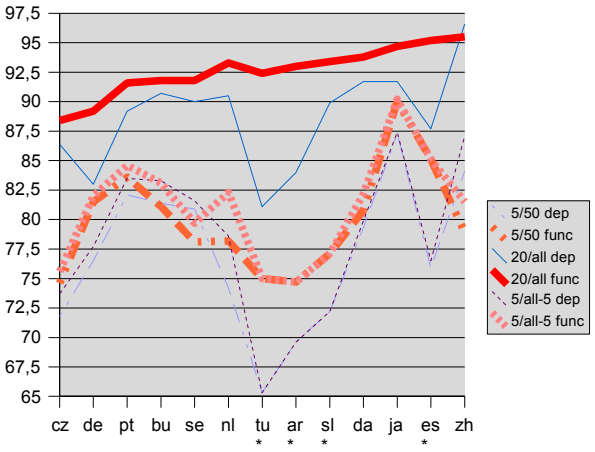


Illustration 2: Performance with different training corpus sizes (upper 2 curves: Test data included)

5 Outlook

We have shown that a probabilistic dependency parser can be built on CG-inspired linguistic principles with a strong focus on function and tag sequences. Given the time constraint and the fact that the learner had to be built from scratch, its performance would encourage further research. In particular, a systematic parameter/performance analysis[13] should be performed for the individual languages. In the long term, a notational harmonization of the treebanks should allow the learner to be seeded with existing hand-written dependency rules.

References

Afonso, S., E. Bick, R. Haber and D. Santos. Floresta Sintá(c)tica: A treebank of Portuguese. In *Proceedings of LREC'02*. pp. 1698-1703 . Paris: ELRA

van der Beek, L. G. Bouma, R. Malouf, G. van Noord. 2002. The Alpino Dependency Treebank. In: *Computational Linguistics in the Netherlands CLIN 2001*. pp. 8-22. Rodopi

Bick, Eckhard. 2005. Turning Constraint Grammar Data into Running Dependency Treebanks. In: Civit, Montserrat & Kübler, Sandra & Martí, Ma. Antònia (ed.), *Proceedings of TLT 2005, Barcelona.* pp.19-2

Brants, S., S. Dipper, S. Hansen, W. Lezius, G. Smith. 2002. The TIGER Treebank. *Proc. of TLT1*, Sozopol

Džerosky, S., T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, A. Žele. 2006. Towards a Slovene Dependency Treebank. In *Proc. of LREC'06*, Genoa

Hajič, J., B. Hladká, and P. Pajas. 2001. The Prague Dependency Treebank: Annotation Structure and Support. In *Proc. of the IRCS Workshop on Linguistic Databases*, pp. 105-114. University of Pennsylvania.

Karlsson, Fred, Atro Vouitilainen, Jukka Heikkilä and A. Anttila. 1995. Constraint Grammar - A Language-Independent System for Parsing Unrestricted Text. Mouton de Gruyter: Berlin.

Kawata, Y. and J. Bartels. 2000. *Stylebook for the Japanese Treebank in VERBMOBIL*. Universität Tübingen: *Verbmobil-Report 240.*

Chen, Keh-Jiann, Chu-Ren Huang, Feng-Yi Chen, Chi-Ching Luo, Ming-Chung Chang, Chao-Jan Chen, and Zhao-Ming Gao. 2003. Sinica Treebank: Design Criteria, Representational Issues and Implementation. In A. Abeille (ed.) *Treebanks Building and Using Parsed Corpora*. Dordrecht:Kluwer, pp231-248.

Kromann, M. T. 2003. The Danish Dependency Treebank. In J. Nivre and E. Hinrichs (ed.) *Proceedings of TLT2003.* Växjö University Press, Sweden

Nart, B. Atalay, Kemal Oflazr, Bilge Say. 2003. The Annotation Process in the Turkish Treebank. In *Proceedings of the EACL Workshop on Linguistically Interpreted Corpora - LINC 2003.* Budapest

Nilsson, J, J. Hall and J. Nivre. 2005. MAMBA Meets TIGER: Reconstructing a Swedish Treebank from Antiquity. In *Proceedings NODALIDA 2005.* Joenssu

Oflazer, K., B. Say, D.Z. Hakkani-Tür, G. Tür. 2003. Building a Turkish Treebank. In A. Abeillé (ed.) *Building and Exploiting Syntactically-annotated Corpora.* Kluwer

Palomar, M. et. al. 2004. Construcción de una base de datos de árboles sintáctico-semánticos para el catalán, euskera y castellano. In: *Proceedings of SEPLN XX*, pp 81-88. Barcelona: ISSN 1135-5948

Simov, K., P. Osenova, A. Simov, M. Kouylekov. 2004. Design and Implementation of the Bulgarian HPSG-based Treebank. In E. Hinrichs and K. Simov (ed.), *Journal of Research on Language and Computation,* Vol. 2, No. 4 , pp. 495-522. Kluwer

Smrž, Otakar, Jan Šnaidauf, and Petr Zemánek. 2002. Prague Dependency Treebank for Arabic: Multi-Level Annotation of Arabic corpus. In *Proceedings of the International Symposium on Processing of Arabic*, pages 147-155, Manouba, Tunisia, April 2002.

[13]Parameters like uniqueness and directedness are already learned by the system (through probability thresholds), while others, like function weights, structural word classes and frequency thresholds for barriers and lexeme n-grams are used now, but with a fixed value for all languages.

Dependency Parsing by Inference over High-recall Dependency Predictions

Sander Canisius, Toine Bogers,
Antal van den Bosch, Jeroen Geertzen
ILK / Computational Linguistics and AI
Tilburg University, P.O. Box 90153,
NL-5000 LE Tilburg, The Netherlands
{S.V.M.Canisius,A.M.Bogers,
Antal.vdnBosch,J.Geertzen}@uvt.nl

Erik Tjong Kim Sang
Informatics Institute
University of Amsterdam, Kruislaan 403
NL-1098 SJ Amsterdam, The Netherlands
erikt@science.uva.nl

1 Introduction

As more and more syntactically-annotated corpora become available for a wide variety of languages, machine learning approaches to parsing gain interest as a means of developing parsers without having to repeat some of the labor-intensive and language-specific activities required for traditional parser development, such as manual grammar engineering, for each new language. The CoNLL-X shared task on multi-lingual dependency parsing (Buchholz et al., 2006) aims to evaluate and advance the state-of-the-art in machine learning-based dependency parsing by providing a standard benchmark set comprising thirteen languages[1]. In this paper, we describe two different machine learning approaches to the CoNLL-X shared task.

Before introducing the two learning-based approaches, we first describe a number of baselines, which provide simple reference scores giving some sense of the difficulty of each language. Next, we present two machine learning systems: 1) an approach that directly predicts all dependency relations in a single run over the input sentence, and 2) a cascade of phrase recognizers. The first approach has been found to perform best and was selected for submission to the competition. We conclude this paper with a detailed error analysis of its output for two of the thirteen languages, Dutch and Spanish.

[1]The data sets were extracted from various existing treebanks (Hajič et al., 2004; Simov et al., 2005; Simov and Osenova, 2003; Chen et al., 2003; Böhmová et al., 2003; Kromann, 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit Torruella and Martí Antonín, 2002; Nilsson et al., 2005; Oflazer et al., 2003; Atalay et al., 2003)

2 Baseline approaches

Given the diverse range of languages involved in the shared task, each having different characteristics probably requiring different parsing strategies, we developed four different baseline approaches for assigning labeled dependency structures to sentences. All of the baselines produce strictly projective structures. While the simple rules implementing these baselines are insufficient for achieving state-of-the-art performance, they do serve a useful role in giving a sense of the difficulty of each of the thirteen languages. The heuristics for constructing the trees and labeling the relations used by each of the four baselines are described below.

Binary right-branching trees The first baseline produces right-branching binary trees. The first token in the sentence is marked as the top node with HEAD `0` and DEPREL `ROOT`. For the rest of the tree, token $n-1$ serves as the HEAD of token n. Figure 1 shows an example of the kind of tree this baseline produces.

Binary left-branching trees The binary left-branching baseline mirrors the previous baseline. The penultimate token in the sentence is marked as the top node with HEAD `0` and DEPREL `ROOT` since punctuation tokens can never serve as `ROOT`[2]. For the rest of the tree, the HEAD of token n is token $n+1$. Figure 2 shows an example of a tree produced by this baseline.

[2]We simply assume the final token in the sentence to be punctuation.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 176–180, New York City, June 2006.

Inward-branching trees In this approach, the first identified verb[3] is marked as the ROOT node. The part of the sentence to the left of the ROOT is left-branching, the part to the right of the ROOT is right-branching. Figure 3 shows an example of a tree produced by this third baseline.

Nearest neighbor-branching trees In our most complex baseline, the first verb is marked as the ROOT node and the other verbs (with DEPREL vc) point to the closest preceding verb. The other tokens point in the direction of their nearest neighboring verb, i.e. the two tokens at a distance of 1 from a verb have that verb as their HEAD, the two tokens at a distance of 2 have the tokens at a distance of 1 as their head, and so on until another verb is a closer neighbor. In the case of ties, i.e. tokens that are equally distant from two different verbs, the token is linked to the preceding token. Figure 4 clarifies this kind of dependency structure in an example tree.

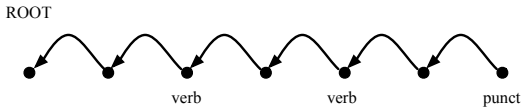


Figure 1: Binary right-branching tree for an example sentence with two verbs.

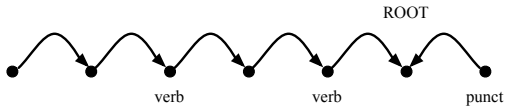


Figure 2: Binary left-branching tree for the example sentence.

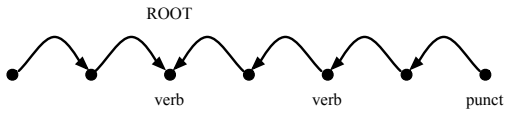


Figure 3: Binary inward-branching tree for the example sentence.

[3]We consider a token a verb if its CPOSTAG starts with a 'V'. This is an obviously imperfect, but language-independent heuristic choice.

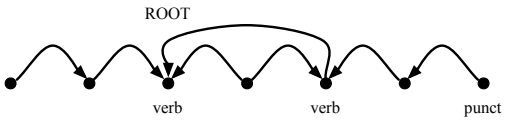


Figure 4: Nearest neighbor-branching tree for the example sentence.

Labeling of identified relations is done using a three-fold back-off strategy. From the training set, we collect the most frequent DEPREL tag for each head-dependent FORM pair, the most frequent DEPREL tag for each FORM, and the most frequent DEPREL tag in the entire training set. The relations are labeled in this order: first, we look up if the FORM pair of a token and its head was present in the training data. If not, then we assign it the most frequent DEPREL tag in the training data for that specific token FORM. If all else fails we label the token with the most frequent DEPREL tag in the entire training set (excluding punct[4] and ROOT).

language	baseline	unlabeled	labeled
Arabic	left	58.82	39.72
Bulgarian	inward	41.29	29.50
Chinese	NN	37.18	25.35
Czech	NN	34.70	22.28
Danish	inward	50.22	36.83
Dutch	NN	34.07	26.87
German	NN	33.71	26.42
Japanese	right	67.18	64.22
Portuguese	right	25.67	22.32
Slovene	right	24.12	19.42
Spanish	inward	32.98	27.47
Swedish	NN	34.30	21.47
Turkish	right	49.03	31.85

Table 1: The labeled and unlabeled scores for the best performing baseline for each language (NN = nearest neighbor-branching).

The best baseline performance (labeled and unlabeled scores) for each language is listed in Table 1. There was no single baseline that outperformed the others on all languages. The nearest neighbor baseline outperformed the other baselines on five of the thirteen languages. The right-branching and

[4]Since the evaluation did not score on punctuation.

inward-branching baselines were optimal on four and three languages respectively. The only language where the left-branching trees provide the best performance is Arabic.

3 Parsing by inference over high-recall dependency predictions

In our approach to dependency parsing, a machine learning classifier is trained to predict (directed) labeled dependency relations between a head and a dependent. For each token in a sentence, instances are generated where this token is a potential dependent of each of the other tokens in the sentence[5]. The label that is predicted for each classification case serves two different purposes at once: 1) it signals whether the token is a dependent of the designated head token, and 2) if the instance does in fact correspond to a dependency relation in the resulting parse of the input sentence, it specifies the type of this relation, as well.

The features we used for encoding instances for this classification task correspond to a rather simple description of the head-dependent pair to be classified. For both the potential head and dependent, there are features encoding a 2-1-2 window of words and part-of-speech tags[6]; in addition, there are two spatial features: a relative position feature, encoding whether the dependent is located to the left or to the right of its potential head, and a distance feature that expresses the number of tokens between the dependent and its head.

One issue that may arise when considering each potential dependency relation as a separate classification case is that inconsistent trees are produced. For example, a token may be predicted to be a dependent of more than one head. To recover a valid dependency tree from the separate dependency predictions, a simple inference procedure is performed. Consider a token for which the dependency relation is to be predicted. For this token, a number of classification cases have been processed, each of them indicating whether and if so how the token is related to one of the other tokens in the sentence. Some of these predictions may be negative, i.e. the token is not a dependent of a certain other token in the sentence, others may be positive, suggesting the token is a dependent of some other token.

If all classifications are negative, the token is assumed to have no head, and consequently no dependency relation is added to the tree for this token; the node in the dependency tree corresponding to this token will then be an isolated one. If one of the classifications is non-negative, suggesting a dependency relation between this token as a dependent and some other token as a head, this dependency relation is added to the tree. Finally, there is the case in which more than one prediction is non-negative. By definition, at most one of these predictions can be correct; therefore, only one dependency relation should be added to the tree. To select the most-likely candidate from the predicted dependency relations, the candidates are ranked according to the classification confidence of the base classifier that predicted them, and the highest-ranked candidate is selected for insertion into the tree.

For our base classifier we used a memory-based learner as implemented by TiMBL (Daelemans et al., 2004). In memory-based learning, a machine learning method based on the nearest-neighbor rule, the class for a given test instance is predicted by performing weighted voting over the class labels of a certain number of most-similar training instances. As a simple measure of confidence for such a prediction, we divide the weight assigned to the majority class by the total weight assigned to all classes. Though this confidence measure is a rather ad-hoc one, which should certainly not be confused with any kind of probability, it tends to work quite well in practice, and arguably did so in the context of this study. The parameters of the memory-based learner have been optimized for accuracy separately for each language on training and development data sampled internally from the training set.

The base classifier in our parser is faced with a classification task with a highly skewed class distribution, i.e. instances that correspond to a dependency relation are largely outnumbered by those that do not. In practice, such a huge number of negative instances usually results in classifiers that tend

[5]To prevent explosion of the number of classification cases to be considered for a sentence, we restrict the maximum distance between a token and its potential head. For each language, we selected this distance so that, on the training data, 95% of the dependency relations is covered.

[6]More specifically, we used the part-of-speech tags from the POSTAG column of the shared task data files.

to predict fairly conservatively, resulting in high precision, but low recall. In the approach introduced above, however, it is better to have high recall, even at the cost of precision, than to have high precision at the cost of recall. A missed relation by the base classifier can never be recovered by the inference procedure; however, due to the constraint that each token can only be a dependent of one head, excessive prediction of dependency relations can still be corrected by the inference procedure. An effective method for increasing the recall of a classifier is down-sampling of the training data. In down-sampling, instances belonging to the majority class (in this case the negative class) are removed from the training data, so as to obtain a more balanced distribution of negative and non-negative instances.

Figure 5 shows the effect of systematically removing an increasingly larger part of the negative instances from the training data. First of all, the figure confirms that down-sampling helps to improve recall, though it does so at the cost of precision. More importantly however, it also illustrates that this improved recall is beneficial for the performance of the dependency parser. The shape of the performance curve of the dependency parser closely follows that of the recall. Remarkably, parsing performance continues to improve with increasingly stronger down-sampling, even though precision drops considerably as a result of this. This shows that the confidence of the classifier for a certain prediction is a sufficiently reliable indication of the quality of that prediction for fixing the over-prediction of dependency relations. Only when the number of negative training instances is reduced to equal the number of positive instances, the performance of the parser is negatively affected. Based on a quick evaluation of various down-sampling ratios on a 90%-10% train-test split of the Dutch training data, we decided to down-sample the training data for all languages with a ratio of two negative instances for each positive one.

Table 2 lists the unlabeled and labeled attachment scores of the resulting system for all thirteen languages.

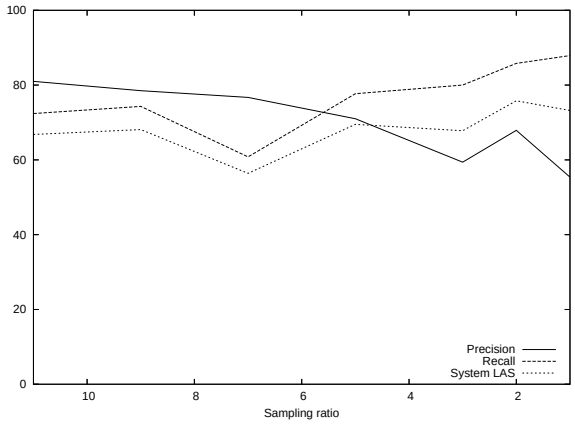


Figure 5: The effect of down-sampling on precision and recall of the base classifier, and on labeled accuracy of the dependency parser. The x-axis refers to the number of negative instances for each positive instance in the training data. Training and testing was performed on a 90%-10% split of the Dutch training data.

4 Cascaded dependency parsing

One of the alternative strategies explored by us was modeling the parsing process as a cascaded pair of basic learners. This approach is similar to Yamada and Matsumoto (2003) but we only use their Left and Right reduction operators, not Shift. In the first phase, each learner predicted dependencies between neighboring words. Dependent words were removed and the remaining words were sent to the learners for further rounds of processing until all words but one had been assigned a head. Whenever crossing links prevented further assignments of heads to words, the learner removed the remaining word requiring the longest dependency link. When the first phase was finished another learner assigned labels to pairs of words present in dependency links.

Unlike in related earlier work (Tjong Kim Sang, 2002), we were unable to compare many different learner configurations. We used two different training files for the first phase: one for predicting the dependency links between adjacent words and one for predicting all other links. As a learner, we used TiMBL with its default parameters. We evaluated different feature sets and ended up with using words, lemmas, POS tags and an extra pair of features with the POS tags of the children of the focus word. With this configuration, this cascaded approach achieved a labeled score of 62.99 on the Dutch test data compared to 74.59 achieved by our main approach.

language	unlabeled	labeled
Arabic	74.59	57.64
Bulgarian	82.51	78.74
Chinese	82.86	78.37
Czech	72.88	60.92
Danish	82.93	77.90
Dutch	77.79	74.59
German	80.01	77.56
Japanese	89.67	87.41
Portuguese	85.61	77.42
Slovene	74.02	59.19
Spanish	71.33	68.32
Swedish	85.08	79.15
Turkish	64.19	51.07

Table 2: The labeled and unlabeled scores for the submitted system for each of the thirteen languages.

5 Error analysis

We examined the system output for two languages in more detail: Dutch and Spanish.

5.1 Dutch

With a labeled attachment score of 74.59 and an unlabeled attachment score of 77.79, our submitted Dutch system performs somewhat above the average over all submitted systems (labeled 70.73, unlabeled 75.07). We review the most notable errors made by our system.

From a part-of-speech (CPOSTAG) perspective, a remarkable relative amount of head and dependency errors are made on **conjunctions**. A likely explanation is that the tag "Conj" applies to both coordinating and subordinating conjunctions; we did not use the FEATS information that made this distinction, which would have likely solved some of these errors.

Left- and right-directed attachment to heads is roughly equally successful. Many errors are made on relations attaching to ROOT; the system appears to be overgenerating attachments to ROOT, mostly in cases when it should have generated rightward attachments. Unsurprisingly, the more distant the head is, the less accurate the attachment; especially recall suffers at distances of three and more tokens.

The most frequent attachment error is generating a ROOT attachment instead of a "mod" (modifier) relation, often occurring at the start of a sentence. Many errors relate to ambiguous adverbs such as *bovendien* (moreover), *tenslotte* (after all), and *zo* (thus), which tend to occur rather frequently at the beginning of sentences in the test set, but less so in the training set. The test set appears to consist largely of formal journalistic texts which typically tend to use these marked rhetorical words in sentence-initial position, while the training set is a more mixed set of texts from different genres plus a significant set of individual sentences, often manually constructed to provide particular examples of syntactic constructions.

5.2 Spanish

The Spanish test data set was the only data set on which the alternative cascaded approach (72.15) outperformed our main approach (68.32). A detailed comparison of the output files of the two systems has revealed two differences. First, the amount of circular links, a pair of words which have each other as head, was larger in the analysis of the submitted system (7%) than in the cascaded analysis (3%) and the gold data (also 3%). Second, the number of root words per sentence (always 1 in the gold data) was more likely to be correct in the cascaded analysis (70% correct; other sentences had no root) than in the submitted approach (40% with 20% of the sentences being assigned no roots and 40% more than one root). Some of these problems might be solvable with post-processing

Acknowledgements

This research is funded by NWO, the Netherlands Organization for Scientific Research under the IMIX programme, and the Dutch Ministry for Economic Affairs' IOP-MMI programme.

References

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of the Tenth Conf. on Computational Natural Language Learning (CoNLL-X)*. SIGNLL.

W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2004. TiMBL: Tilburg memory based learner, version 5.1, reference guide. Technical Report ILK 04-02, ILK Research Group, Tilburg University.

Erik Tjong Kim Sang. 2002. Memory-based shallow parsing. *Journal of Machine Learning Research*, 2(Mar):559–594.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *8th International Workshop of Parsing Technologies (IWPT2003)*. Nancy, France.

Projective Dependency Parsing with Perceptron

Xavier Carreras, Mihai Surdeanu and **Lluís Màrquez**
TALP Research Centre – Software Department (LSI)
Technical University of Catalonia (UPC)
Campus Nord - Edifici Omega, Jordi Girona Salgado 1–3, E-08034 Barcelona
{carreras,surdeanu,lluism}@lsi.upc.edu

Abstract

We describe an online learning dependency parser for the CoNLL-X Shared Task, based on the bottom-up projective algorithm of Eisner (2000). We experiment with a large feature set that models: the tokens involved in dependencies and their immediate context, the surface-text distance between tokens, and the syntactic context dominated by each dependency. In experiments, the treatment of multilingual information was totally blind.

1 Introduction

We describe a learning system for the CoNLL-X Shared Task on multilingual dependency parsing (Buchholz et al., 2006), for 13 different languages.

Our system is a bottom-up projective dependency parser, based on the cubic-time algorithm by Eisner (1996; 2000). The parser uses a learning function that scores all possible labeled dependencies. This function is trained globally with online Perceptron, by parsing training sentences and correcting its parameters based on the parsing mistakes. The features used to score, while based on the previous work in dependency parsing (McDonald et al., 2005), introduce some novel concepts such as better codification of context and surface distances, and runtime information from dependencies previously parsed.

Regarding experimentation, the treatment of multilingual data has been totally blind, with no special processing or features that depend on the language. Considering its simplicity, our system achieves moderate but encouraging results, with an overall labeled attachment accuracy of 74.72% on the CoNLL-X test set.

2 Parsing and Learning Algorithms

This section describes the three main components of the dependency parsing: the parsing model, the parsing algorithm, and the learning algorithm.

2.1 Model

Let $1, \ldots, L$ be the dependency labels, defined beforehand. Let x be a sentence of n words, $x_1 \ldots x_n$. Finally, let $\mathcal{Y}(x)$ be the space of well-formed dependency trees for x. A dependency tree $y \in \mathcal{Y}(x)$ is a set of n dependencies of the form $[h, m, l]$, where h is the index of the head word ($0 \leq h \leq n$, where 0 means root), m is the index of the modifier word ($1 \leq m \leq n$), and l is the dependency label ($1 \leq l \leq L$). Each word of x participates as a modifier in exactly one dependency of y.

Our dependency parser, dp, returns the maximum scored dependency tree for a sentence x:

$$\mathrm{dp}(x, \mathbf{w}) = \arg\max_{y \in \mathcal{Y}(x)} \sum_{[h,m,l] \in y} \mathrm{sco}([h, m, l], x, y, \mathbf{w})$$

In the formula, $\mathbf{w}$ is the weight vector of the parser, that is, the set of parameters used to score dependencies during the parsing process. It is formed by a concatenation of L weight vectors, one for each dependency label, $\mathbf{w} = (\mathbf{w}^1, \ldots, \mathbf{w}^l, \ldots, \mathbf{w}^L)$. We assume a feature extraction function, ϕ, that represents an unlabeled dependency $[h, m]$ in a vector of D features. Each of the $\mathbf{w}^l$ has D parameters or dimensions, one for each feature. Thus, the global

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 181–185, New York City, June 2006.

weight vector $\mathbf{w}$ maintains $L \times D$ parameters. The scoring function is defined as follows:

$$\mathrm{sco}([h, m, l], x, y, \mathbf{w}) = \phi(h, m, x, y) \cdot \mathbf{w}^l$$

Note that the scoring of a dependency makes use of y, the tree that contains the dependency. As described next, at scoring time y just contains the dependencies found between h and m.

2.2 Parsing Algorithm

We use the cubic-time algorithm for dependency parsing proposed by Eisner (1996; 2000). This parsing algorithm assumes that trees are projective, that is, dependencies never cross in a tree. While this assumption clearly does not hold in the CoNLL-X data (only Chinese trees are actually 100% projective), we chose this algorithm for simplicity. As it will be shown, the percentage of non-projective dependencies is not very high, and clearly the error rates we obtain are caused by other major factors.

The parser is a bottom-up dynamic programming algorithm that visits sentence spans of increasing length. In a given span, from word s to word e, it completes two partial dependency trees that cover all words within the span: one rooted at s and the other rooted at e. This is done in two steps. First, the optimal dependency structure internal to the span is chosen, by combining partial solutions from internal spans. This structure is completed with a dependency covering the whole span, in two ways: from s to e, and from e to s. In each case, the scoring function is used to select the dependency label that maximizes the score.

We take advantage of this two-step processing to introduce features for the scoring function that represent *some* of the internal dependencies of the span (see Section 3 for details). It has to be noted that the parsing algorithm we use does not score dependencies on top of every possible internal structure. Thus, by conditioning on features extracted from y we are making the search approximative.

2.3 Perceptron Learning

As learning algorithm, we use Perceptron tailored for structured scenarios, proposed by Collins (2002). In recent years, Perceptron has been used in a number of Natural Language Learning works, such as in partial parsing (Carreras et al., 2005) or even dependency parsing (McDonald et al., 2005).

```
w = 0
for t = 1 to T
    foreach training example (x, y) do
        ŷ = dp(x, w)
        foreach [h, m, l] ∈ y \ ŷ do
            w^l = w^l + φ(h, m, x, ŷ)
        foreach [h, m, l] ∈ ŷ \ y do
            w^l = w^l − φ(h, m, x, ŷ)
return w
```

Figure 1: Pseudocode of the Perceptron Algorithm. T is a parameter that indicates the number of epochs that the algorithm cycles the training set.

Perceptron is an online learning algorithm that learns by correcting mistakes made by the parser when visiting training sentences. The algorithm is extremely simple, and its cost in time and memory is independent from the size of the training corpora. In terms of efficiency, though, the parsing algorithm must be run at every training sentence.

Our system uses the regular Perceptron working in primal form. Figure 1 sketches the code. Given the number of languages and dependency types in the CoNLL-X exercise, we found prohibitive to work with a dual version of Perceptron, that would allow the use of a kernel function to expand features.

3 Features

The feature extraction function, $\phi(h, m, x, y)$, represents in a feature vector a dependency from word positions m to h, in the context of a sentence x and a dependency tree y. As usual in discriminative learning, we work with binary indicator features: if a certain feature is observed in an instance, the value of that feature is 1; otherwise, the value is 0. For convenience, we describe ϕ as a composition of several base feature extraction functions. Each extracts a number of disjoint features. The feature extraction function $\phi(h, m, x, y)$ is calculated as:

$$\begin{aligned}
&\phi_{token}(x, h, \text{“}head\text{”}) \; + \; \phi_{tctx}(x, h, \text{“}head\text{”}) \; + \\
&\phi_{token}(x, m, \text{“}mod\text{”}) \; + \; \phi_{tctx}(x, m, \text{“}mod\text{”}) \; + \\
&\phi_{dep}(x, mmd_{h,m}) \; + \; \phi_{dctx}(x, mmd_{h,m}) \; + \\
&\phi_{dist}(x, mmd_{h,m}) \; + \; \phi_{runtime}(x, y, h, m, d_{h,m})
\end{aligned}$$

where ϕ_{token} extracts context-independent token features, ϕ_{tctx} computes context-based token features, ϕ_{dep} computes context-independent depen-

$\phi_{\mathbf{token}}(\mathbf{x}, \mathbf{i}, \mathbf{type})$
$type \cdot w(x_i)$
$type \cdot l(x_i)$
$type \cdot cp(x_i)$
$type \cdot fp(x_i)$
foreach(ms): $type \cdot ms(x_i)$
$type \cdot w(x_i) \cdot cp(x_i)$
foreach(ms): $type \cdot w(x_i) \cdot ms(x_i)$
$\phi_{\mathbf{tctx}}(\mathbf{x}, \mathbf{i}, \mathbf{type})$
$\phi_{token}(x, i-1, type \cdot string(i-1))$
$\phi_{token}(x, i-2, type \cdot string(i-2))$
$\phi_{token}(x, i+1, type \cdot string(i+1))$
$\phi_{token}(x, i+2, type \cdot string(i+2))$
$type \cdot cp(x_i) \cdot cp(x_{i-1})$
$type \cdot cp(x_i) \cdot cp(x_{i-1}) \cdot cp(x_{i-2})$
$type \cdot cp(x_i) \cdot cp(x_{i+1})$
$type \cdot cp(x_i) \cdot cp(x_{i+1}) \cdot cp(x_{i+2})$

Table 1: Token features, both context-independent (ϕ_{token}) and context-based (ϕ_{tctx}). $type$ - token type, i.e. "head" or "mod", w - token word, l - token lemma, cp - token coarse part-of-speech (POS) tag, fp - token fine-grained POS tag, ms - token morpho-syntactic feature. The $\cdot$ operator stands for string concatenation.

$\phi_{\mathbf{dep}}(\mathbf{x}, \mathbf{i}, \mathbf{j}, \mathbf{dir})$
$dir \cdot w(x_i) \cdot cp(x_i) \cdot w(x_j) \cdot cp(x_j)$
$dir \cdot cp(x_i) \cdot w(x_j) \cdot cp(x_j)$
$dir \cdot w(x_i) \cdot w(x_j) \cdot cp(x_j)$
$dir \cdot w(x_i) \cdot cp(x_i) \cdot cp(x_j)$
$dir \cdot w(x_i) \cdot cp(x_i) \cdot w(x_j)$
$dir \cdot w(x_i) \cdot w(x_j)$
$dir \cdot cp(x_i) \cdot cp(x_j)$
$\phi_{\mathbf{dctx}}(\mathbf{x}, \mathbf{i}, \mathbf{j}, \mathbf{dir})$
$dir \cdot cp(x_i) \cdot cp(x_{i+1}) \cdot cp(x_{j-1}) \cdot cp(x_j)$
$dir \cdot cp(x_{i-1}) \cdot cp(x_i) \cdot cp(x_{j-1}) \cdot cp(x_j)$
$dir \cdot cp(x_i) \cdot cp(x_{i+1}) \cdot cp(x_j) \cdot cp(x_{j+1})$
$dir \cdot cp(x_{i-1}) \cdot cp(x_i) \cdot cp(x_j) \cdot cp(x_{j+1})$

Table 2: Dependency features, both context-independent (ϕ_{dep}) and context-based (ϕ_{dctx}), between two points i and j, $i < j$. dir - dependency direction: left to right or right to left.

dency features, ϕ_{dctx} extracts contextual dependency features, ϕ_{dist} calculates surface-distance features between the two tokens, and finally, $\phi_{runtime}$ computes dynamic features at runtime based on the dependencies previously built for the given interval during the bottom-up parsing. $mmd_{h,m}$ is a shorthand for a triple of numbers: $\min(h, m)$, $\max(h, m)$ and $d_{h,m}$ (a sign indicating the direction, i.e., $+1$ if $m < h$, and -1 otherwise).

We detail the token features in Table 1, the dependency features in Table 2, and the surface-distance features in Table 3. Most of these features are inspired by previous work in dependency parsing (McDonald et al., 2005; Collins, 1999). What is important for the work presented here is that we construct explicit feature combinations (see above tables) because we configured our linear predictors in primal form, in order to keep training times reasonable.

$\phi_{\mathbf{dist}}(\mathbf{x}, \mathbf{i}, \mathbf{j}, \mathbf{dir})$
foreach(k $\in (i, j)$): $dir \cdot cp(x_i) \cdot cp(x_k) \cdot cp(x_j)$
number of tokens between i and j
number of verbs between i and j
number of coordinations between i and j
number of punctuations signs between i and j

Table 3: Surface distance features between points i and j. Numeric features are discretized using "binning" to a small number of intervals.

$\phi_{\mathbf{runtime}}(\mathbf{x}, \mathbf{y}, \mathbf{h}, \mathbf{m}, \mathbf{dir})$
let $l_1, \ldots, l_S$ be the labels of dependencies in y that attach to h and are found from m to h.
foreach i, $1 \leq i \leq S$: $dir \cdot cp(x_h) \cdot cp(x_m) \cdot l_i$
if $S \geq 1$, $dir \cdot cp(x_h) \cdot cp(x_m) \cdot l_1$
if $S \geq 2$, $dir \cdot cp(x_h) \cdot cp(x_m) \cdot l_1 \cdot l_2$
if $S \geq 3$, $dir \cdot cp(x_h) \cdot cp(x_m) \cdot l_1 \cdot l_2 \cdot l_3$
if $S \geq 4$, $dir \cdot cp(x_h) \cdot cp(x_m) \cdot l_1 \cdot l_2 \cdot l_3 \cdot l_4$
if $S = 0$, $dir \cdot cp(x_h) \cdot cp(x_m) \cdot null$
if $0 < S \leq 4$, $dir \cdot cp(x_h) \cdot cp(x_m) \cdot regular$
if $S > 4$, $dir \cdot cp(x_h) \cdot cp(x_m) \cdot big$

Table 4: Runtime features of y between m and h.

While the features presented in Tables 1, 2, and 3 are straightforward exploitations of the training data, the runtime features ($\phi_{runtime}$) take a different, and to our knowledge novel in the proposed framework, approach: for a dependency from m to h, they represent the dependencies found between m and h that attach also to h. They are described in detail in Table 4. As we have noted above, these features are possible because of the parsing scheme, which scores a dependency only after all dependencies spanned by it are scored.

4 Experiments and Results

We experimented on the 13 languages proposed in the CoNLL-X Shared Task (Hajič et al., 2004; Simov et al., 2005; Simov and Osenova, 2003; Chen et al., 2003; Böhmová et al., 2003; Kromann, 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit and Martí, 2002; Nilsson et al., 2005; Oflazer et al., 2003; Atalay et al., 2003). Our approach to deal with many different languages was totally blind: we did not inspect the data to motivate language-specific features or processes.

We did feature filtering based on frequency counts. Our feature extraction patterns, that exploit both lexicalization and combination, generate millions of feature dimensions, even with small datasets. Our criterion was to use at most 500,000 different dimensions in each label weight vector. For each language, we generated all possible features, and then filtered out most of them according to the counts. Depending on the number of training sentences, our counts cut-offs vary from 3 to 15.

For each language, we held out from training data a portion of sentences (300, 500 or 1000 depending on the total number of sentences) and trained a model for up to 20 epochs in the rest of the data. We evaluated each model on the held out data for different number of training epochs, and selected the optimum point. Then, we retrained each model on the whole training set for the selected number of epochs.

Table 5 shows the attachment scores obtained by our system, both unlabeled (UAS) and labeled (LAS). The first column (GOLD) presents the LAS obtained with a perfect scoring function: the loss in accuracy is related to the projectivity assumption of our parsing algorithm. Dutch turns out to be the most non-projective language, with a loss in accuracy of 5.44%. In our opinion, the loss in other languages is relatively small, and is not a major limitation to achieve a high performance in the task. Our system achieves an overall LAS of 74.72%, with substantial variation from one language to another. Turkish, Arabic, Dutch, Slovene and Czech turn out to be the most difficult languages for our system, with accuracies below 70%. The easiest language is clearly Japanese, with a LAS of 88.13%, followed by Chinese, Portuguese, Bulgarian and German, all with LAS above 80%.

Table 6 shows the contribution of base feature extraction functions. For four languages, we trained models that increasingly incorporate base functions. It can be shown that all functions contribute to a better score. Contextual features (ϕ_3) bring the system to the final order of performance, while distance (ϕ_4) and runtime (ϕ) features still yield substantial improvements.

5 Analysis and Conclusions

It is difficult to explain the difference in performance across languages. Nevertheless, we have identified four generic factors that we believe caused the most errors across all languages:

	GOLD	UAS	LAS
Bulgarian	99.56	88.81	**83.30**
Arabic	99.76	72.65	**60.94**
Chinese	100.0	88.65	**83.68**
Czech	97.78	77.44	**68.82**
Danish	99.18	85.67	**79.74**
Dutch	94.56	71.39	**67.25**
German	98.84	85.90	**82.41**
Japanese	99.16	90.79	**88.13**
Portuguese	98.54	87.76	**83.37**
Slovene	98.38	77.72	**68.43**
Spanish	99.96	80.77	**77.16**
Swedish	99.64	85.54	**78.65**
Turkish	98.41	70.05	**58.06**
Overall	98.68	81.19	**74.72**

Table 5: Results of the system on test data. GOLD: labeled attachment score using gold scoring functions; the loss in accuracy is caused by the projectivity assumption made by the parser. UAS : unlabeled attachment score. LAS : labeled attachment score, the measure to compare systems in CoNLL-X. Bulgarian is excluded from overall scores.

	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ
Turkish	33.02	48.00	55.33	57.16	58.06
Spanish	12.80	53.80	68.18	74.27	77.16
Portuguese	47.10	64.74	80.89	82.89	83.37
Japanese	38.78	78.13	86.87	88.27	88.13

Table 6: Labeled attachment scores at increasing feature configurations. ϕ_1 uses only ϕ_{token} at the head and modifier. ϕ_2 extends ϕ_1 with ϕ_{dep}. ϕ_3 incorporates context features, namely ϕ_{tctx} at the head and modifier, and ϕ_{dctx}. ϕ_4 extends ϕ_3 with ϕ_{dist}. Finally, the final feature extraction function ϕ increases ϕ_4 with $\phi_{runtime}$.

Size of training sets: the relation between the amount of training data and performance is strongly supported in learning theory. We saw the same relation in this evaluation: for Turkish, Arabic, and Slovene, languages with limited number of training sentences, our system obtains accuracies below 70%. However, one can not argue that the training size is the only cause of errors: Czech has the largest training set, and our accuracy is also below 70%.

Modeling large distance dependencies: even though we include features to model the distance between two dependency words (ϕ_{dist}), our analysis indicates that these features fail to capture all the intricacies that exist in large-distance dependencies. Table 7 shows that, for the two languages analyzed, the system performance decreases sharply as the distance between dependency tokens increases.

	to root	1	2	$3-6$	$>=7$
Spanish	83.04	93.44	86.46	69.97	61.48
Portuguese	90.81	96.49	90.79	74.76	69.01

Table 7: $F_{\beta=1}$ score related to dependency token distance.

	Fully lexicalized	One token unlexicalized	Fully unlexicalized
Spanish	34.80%	54.77%	10.43%
Portuguese	42.94%	49.26%	7.80%

Table 8: Degree of dependency lexicalization.

Modeling context: many attachment decisions, e.g. prepositional attachment, depend on additional context outside of the two dependency tokens. To address this issue, we have included in our model features to capture context, both static (ϕ_{dctx} and ϕ_{tctx}) and dynamic ($\phi_{runtime}$). Nevertheless, our error analysis indicates that our model is not rich enough to capture the context required to address complex dependencies. All the top 5 focus words with the majority of errors for Spanish and Portuguese – "y", "de", "a", "en", and "que" for Spanish, and "em", "de", "a", "e", and "para" for Portuguese – indicate complex dependencies such as prepositional attachments or coordinations.

Projectivity assumption: Dutch is the language with most crossing dependencies in this evaluation, and the accuracy we obtain is below 70%.

On the Degree of Lexicalization We conclude the error analysis of our model with a look at the degree of lexicalization in our model. A quick analysis of our model on the test data indicates that only 34.80% of the dependencies for Spanish and 42.94% of the dependencies for Portuguese are fully lexicalized, i.e. both the head and modifier words appear in the model feature set (see Table 8). There are two reasons that cause our model to be largely unlexicalized: (a) in order to keep training times reasonable we performed heavy filtering of all features based on their frequency, which eliminates many lexicalized features from the final model, and (b) due to the small size of most of the training corpora, most lexicalized features simply do not appear in the testing section. Considering these results, a reasonable question to ask is: how much are we losing because of this lack of lexical information? We give an approximate answer by analyzing the percentage of fully-lexicalized dependencies that are correctly parsed by our model. Assuming that our model scales well, the accuracy on fully-lexicalized dependencies is an indication for the gain (or loss) to be had from lexicalization. Our model parses fully-lexicalized dependencies with an accuracy of 74.81% LAS for Spanish (2.35% *lower* than the overall score) and of 83.77% LAS for Portuguese (0.40% higher than the overall score). This analysis indicates that our model has limited gains (if any) from lexicalization.

In order to improve the quality of our dependency parser we will focus on previously reported issues that can be addressed by a parsing model: large-distance dependencies, better modeling of context, and non-projective parsing algorithms.

Acknowledgements

This work was partially funded by the European Union Commission (PASCAL - IST-2002-506778) and Spanish Ministry of Science and Technology (TRANGRAM - TIN2004-07925-C03-02). Mihai Surdeanu was supported by a Ramon y Cajal fellowship of the latter institution.

References

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of the Tenth Conf. on Computational Natural Language Learning (CoNLL-X)*. SIGNLL.

X. Carreras, Lluís Màrquez, and J. Castro. 2005. Filtering-ranking perceptron learning for partial parsing. *Machine Learning*, 1–3(60):41–71.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP-2002*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of the 16th Intern. Conf. on Computational Linguistics (COLING)*.

J. Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In H. C. Bunt and A. Nijholt, editors, *New Developments in Natural Language Parsing*, pages 29–62. Kluwer Academic Publishers.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of the 43rd Annual Meeting of the ACL*.

A Pipeline Model for Bottom-Up Dependency Parsing

Ming-Wei Chang **Quang Do** **Dan Roth**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{mchang21, quangdo2, danr}@uiuc.edu

Abstract

We present a new machine learning framework for multi-lingual dependency parsing. The framework uses a linear, pipeline based, bottom-up parsing algorithm, with a look ahead local search that serves to make the local predictions more robust. As shown, the performance of the first generation of this algorithm is promising.

1 System Description

1.1 Parsing as a Pipeline

Pipeline computation is a common computational strategy in natural language processing, where a task is decomposed into several stages that are solved sequentially. For example, a semantic role labeling program may start by using a part-of-speech tagger, than apply a shallow parser to chunk the sentence into phrases, and continue by identifying predicates and arguments and then classifying them.

(Yamada and Matsumoto, 2003) proposed a bottom-up dependency parsing algorithm, where the local actions, chosen from among *Shift, Left, Right*, are used to generate a dependency tree using a shift-reduce parsing approach. Moreover, they used SVMs to learn the parsing decisions between pairs of consecutive words in the sentences [1]. This is a true pipeline approach in that the classifiers are trained on individual decisions rather than on the overall quality of the parser, and chained to yield the global structure. It suffers from the limitations of pipeline processing, such as accumulation of errors, but nevertheless, yields very competitive parsing results.

We devise two natural principles for enhancing pipeline models. First, inference procedures should be incorporated to make robust prediction for each stage. Second, the number of predictions should be minimized to prevent error accumulation. According to these two principles, we propose an improved pipeline framework for multi-lingual dependency parsing that aims at addressing the limitations of the pipeline processing. Specifically, (1) we use local search, a look ahead policy, to improve the accuracy of the predicted actions, and (2) we argue that the parsing algorithm we used minimizes the number of actions (Chang et al., 2006).

We use the set of actions: *Shift, Left, Right, WaitLeft, WaitRight* for the parsing algorithm. The pure *Wait* action was suggested in (Yamada and Matsumoto, 2003). However, here we come up with these five actions by separating actions *Left* into (real) *Left* and *WaitLeft*, and *Right* into (real) *Right* and *WaitRight*. Predicting these turns out to be easier due to finer granularity. We then use local search over consecutive actions and better exploit the dependencies among them.

The parsing algorithm is a modified shift-reduce parser (Aho et al., 1986) that makes use of the actions described above and applies them in a left to right manner on consecutive word pairs (a, b) $(a < b)$ in the word list T. T is initialized as the full sentence. Latter, the actions will change the contents of T. The actions are used as follows:

[1] A pair of words may become consecutive after the words between them become the children of these two words

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 186–190, New York City, June 2006.

S*hift*: there is no relation between a and b.

R*ight*: b is the parent of a,

L*eft*: a is the parent of b

W*ait***L***eft*: a is the parent of b, but it's possible that b is a parent of other nodes. Action is deferred.

The actions control the procedure of building trees. When **L***eft* or **R***ight* is performed, the algorithm has found a parent and a child. Then, the function *deleteWord* will be called to eliminate the child word, and the procedure will be repeated until the tree is built. In projective languages, we discovered that action **W***ait***R***ight* is not needed. Therefore, for projective languages, we just need 4 actions.

In order to complete the description of the algorithm we need to describe which pair of consecutive words to consider once an action is taken. We describe it via the notion of the *focus point*, which represents the index of the current word in T. In fact, determining the focus point does not affect the correctness of the algorithm. It is easy to show that any pair of consecutive words in the sentence can be considered next. If the correct action is chosen for the corresponding pair, this will eventually yield the correct tree (but may necessitate multiple cycles through the sentence).

In practice, however, the actions chosen will be noisy, and a wasteful focus point policy will result in a large number of actions, and thus in error accumulation. To minimize the number of actions taken, we want to find a good focus point placement policy.

There are many natural placement policies that we can consider (Chang et al., 2006). In this paper, according to the policy we used, after **S** and **WL**, the focus point moves one word to the right. After **L** or **R**, we adopt the policy *Step Back*: the focus moves back one word to the left. Although the focus placement policy here is similar to (Yamada and Matsumoto, 2003), they did not explain why they made this choice. In (Chang et al., 2006), we show that the policy movement used here minimized the number of actions during the parsing procedure. We can also show that the algorithm can parse a sentence with projective relationships in only one round.

Once the parsing algorithm, along with the focus point policy, is determined, we can train the action classifiers. Given an annotated corpus, the parsing algorithm is used to determine the action taken for each consecutive pair; this is used to train a classifier to predict one of the four actions. The details of the classifier and the features are given in Section 3.

Algorithm 1 Pseudo Code of the dependency parsing algorithm. *getFeatures* extracts the features describing the currently considered pair of words; *getAction* determines the appropriate action for the pair; *assignParent* assigns the parent for the child word based on the action; and *deleteWord* deletes the word which become child once the action is taken.

```
Let t represents for a word and its part of speech
For sentence T = {t_1, t_2, ..., t_n}
focus= 1
while focus< |T| do
  v = getFeatures(t_focus, t_focus+1)
  α = getAction(t_focus, t_focus+1, v)
  if α = L or α = R then
    assignParent(t_focus, t_focus+1, α)
    deleteWord(T, focus, α)
    // performing Step Back here
    focus = focus − 1
  else
    focus = focus + 1
  end if
end while
```

When we apply the trained model on new data, the sentence is processed from left to right to produce the predicted dependency tree. The evaluation process is somewhat more involved, since the action classifier is not used as it is, but rather via a local search inference step. This is described in Section 2. Algorithm 1 depicts the pseudo code of our parsing algorithm.

Our algorithm is designed for projective languages. For non-projective relationships in some languages, we convert them into near projective ones. Then, we directly apply the algorithm on modified data in training stage. Because the sentences in some language, such as Czech, etc. , may have multi roots, in our experiment, we ran multiple rounds of Algorithm 1 to build the tree.

1.2 Labeling the Type of Dependencies

In our work, labeling the type of dependencies is a post-task after the phase of predicting the head for the tokens in the sentences. This is a multi-class classification task. The number of the de-

pendency types for each language can be found in the organizer's introduction paper of the shared task of CoNLL-X. In the phase of learning dependency types, the parent of the tokens, which was labeled in the first phase, will be used as features. The predicted actions can help us to make accurate predictions for dependency types.

1.3 Dealing with Crossing Edges

The algorithm described in previous section is primarily designed for projective languages. To deal with non-projective languages, we use a similar approach of (Nivre and Nilsson, 2005) to map non-projective trees to projective trees. Any single rooted projective dependency tree can be mapped into a projective tree by the *Lift* operation. The definition of *Lift* is as follows: *Lift*$(w_j \rightarrow w_k) =$ **parent**$(w_j) \rightarrow w_k$, where $a \rightarrow b$ means that a is the parent of b, and **parent** is a function which returns the parent word of the given word. The procedure is as follows. First, the mapping algorithm examines if there is a crossing edge in the current tree. If there is a crossing edge, it will perform *Lift* and replace the edge until the tree becomes projective.

2 Local Search

The advantage of a pipeline model is that it can use more information that is taken from the outcomes of previous prediction. However, this may result in accumulating error. Therefore, it is essential for our algorithm to use a reliable action predictor. This motivates the following approach for making the local prediction in a pipeline model more reliable. Informally, we devise a local search algorithm and use it as a look ahead policy, when determining the predicted action.

In order to improve the accuracy, we might want to examine all the combinations of actions proposed and choose the one that maximizes the score. It is clearly intractable to find the global optimal prediction sequence in a pipeline model of the depth we consider. The size of the possible action sequence increases exponentially so that we can not examine every possibility. Therefore, a local search framework which uses additional information, however, is suitable and tractable.

The local search algorithm is presented in Algorithm 2. The algorithm accepts two parameters, *model* and *depth*. We assume a classifier that can give a confidence in its prediction. This is represented here by *model*. *depth* is the parameter determining the depth of the local search. *State* encodes the configuration of the environment (in the context of the dependency parsing this includes the sentence, the focus point and the current parent and children for each node). Note that the features extracted for the action classifier depends on *State*, and *State* changes by the *update* function when a prediction is made. In this paper, the *update* function cares about the child word elimination, relationship addition and *focus point* movement.

The search algorithm will perform a search of length *depth*. Additive scoring is used to score the sequence, and the first action in this sequence is performed. Then, the *State* is updated, determining the

Algorithm 2 Pseudo code for the local search algorithm. In the algorithm, $\mathbf{y}$ represents the a action sequence. The function *search* considers all possible action sequences with $|depth|$ actions and returns the sequence with highest score.

```
Algo predictAction(model, depth, State)
x = getNextFeature(State)
y = search(x, depth, model, State)
lab = y[1]
State = update(State, lab)
return lab

Algo search(x, depth, model, State)
maxScore = −∞
F = {y | ||y|| = depth}
for y in F do
  s = 0, TmpState = State
  for i = 1 ... depth do
    x = getNextFeature(TmpState)
    s = s + log(score(y[i], x))
    TmpState = update(TmpState, y[i])
  end for
  if s > maxScore then
    ŷ = y
    maxScore = s
  end if
end for
return ŷ
```

next features for the action classifiers and *search* is called again.

One interesting property of this framework is that we use future information in addition to past information. The pipeline model naturally allows access to all the past information. But, since our algorithm uses the search as a look ahead policy, it can produce more robust results.

3 Experiments and Results

In this work we used as our learning algorithm a regularized variation of the perceptron update rule as incorporated in SNoW (Roth, 1998; Carlson et al., 1999), a multi-class classifier that is specifically tailored for large scale learning tasks. SNoW uses softmax over the raw activation values as its confidence measure, which can be shown to be a reliable approximation of the labels' probabilities. This is used both for labeling the actions and types of dependencies. There is no special language enhancement required for each language. The resources provided for 12 languages are described in: (Hajič et al., 2004; Chen et al., 2003; Böhmová et al., 2003; Kromann, 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit Torruella and Martí Antonín, 2002; Nilsson et al., 2005; Oflazer et al., 2003; Atalay et al., 2003).

3.1 Experimental Setting

The feature set plays an important role in the quality of the classifier. Basically, we used the same feature set for the action selection classifiers and for the label classifiers. In our work, each example has average fifty active features. For each word pair (w_1, w_2), we used their LEMMA, the POSTAG and also the POSTAG of the children of w_1 and w_2. We also included the LEMMA and POSTAG of surrounding words in a window of size $(2, 4)$. We considered 2 words before w_1 and 4 words after w_2 (we agree with the window size in (Yamada and Matsumoto, 2003)). The major difference of our feature set compared with the one in (Yamada and Matsumoto, 2003) is that we included the previous predicted action. We also added some conjunctions of the above features to ensure expressiveness of the model. (Yamada and Matsumoto, 2003) made use of the polynomial kernel of degree 2 so they in fact use more conjunctive features. Beside these features, we incorporated the information of FEATS for the languages when it is available. The columns in the data files we used for our work are the LEMMA, POSTAG, and the FEATS, which is treated as atomic. Due to time limitation, we did not apply the local search algorithm for the languages having the FEATS features.

3.2 Results

Table 1 shows our results on Unlabeled Attachment Scores (UAS), Labeled Attachment Scores (LAS), and Label Accuracy score (LAC) for 12 languages. Our results are compared with the average scores (AV) and the standard deviations (SD), of all the systems participating in the shared task of CoNLL-X.

Our average UAS for 12 languages is 83.54% with the standard deviation 6.01; and 76.80% with the standard deviation 9.43 for average LAS.

4 Analysis and Discussion

We observed that our UAS for Arabic is generally lower than for other languages. The reason for the low accuracy of Arabic is that the sentence is very long. In the training data for Arabic, there are 25% sentences which have more than 50 words. Since we use a pipeline model in our algorithm, it required more predictions to complete a long sentence. More predictions in pipeline models may result in more mistakes. We think that this explains our relatively low Arabic result. Moreover, in our current system, we use the same window size (2,4) for feature extraction in all languages. Changing the windows size seems to be a reasonable step when the sentences are longer.

For Czech, one reason for our relatively low result is that we did not use the whole training corpus due to time limitation [2] . Actually, in our experiment on the development set, when we increase the size of training data in the training phase we got significantly higher result than the system trained on the smaller data. The other problem for Czech is that Czech is one of the languages with many types of part of speech and dependency types, and also the

[2]Training our system for most languages takes 30 minutes or 1 hour for both phases of labeling HEAD and DEPREL. It takes 6-7 hours for Czech with 50% training data.

Language	UAS			LAS			LAC		
	Ours	AV	SD	Ours	AV	SD	Ours	AV	SD
Arabic	76.09	73.48	4.94	60.92	59.94	6.53	75.69	75.12	5.49
Chinese	89.60	84.85	5.99	85.05	78.32	8.82	87.28	81.66	7.92
Czech	81.78	77.01	6.70	72.88	67.17	8.93	80.42	76.59	7.69
Danish	86.85	84.52	8.97	80.60	78.31	11.34	86.51	84.50	4.35
Dutch	76.25	75.07	5.78	72.91	70.73	6.66	80.15	77.57	5.92
German	86.90	82.60	6.73	84.17	78.58	7.51	91.03	86.26	6.01
Japanese	90.77	89.05	5.20	89.07	85.86	7.09	92.18	89.90	5.36
Portuguese	88.60	86.46	4.17	83.99	80.63	5.83	88.84	85.35	5.45
Slovene	80.32	76.53	4.67	69.52	65.16	6.78	79.26	76.31	6.40
Spanish	83.09	77.76	7.81	79.72	73.52	8.41	89.26	85.71	4.56
Swedish	89.05	84.21	5.45	82.31	76.44	6.46	84.82	80.00	6.24
Turkish	73.15	69.35	5.51	60.51	55.95	7.71	73.75	69.59	7.94

Table 1: Our results are compared with the average scores. UAS=Unlabeled Attachment Score, LAS=Labeled Attachment Score, LAC=Label Accuracy, AV=Average score, and SD=standard deviation.

length of the sentences in Czech is relatively long. These facts make recognizing the HEAD and the types of dependencies more difficult.

Another interesting aspect is that we have not used the information about the syntactic and/or morphological features (FEATS) properly. For the languages for which FEATS is available, we have a larger gap, compared with the top system.

5 Further Work and Conclusion

In the shared task of CoNLL-X, we have shown that our dependency parsing system can do well on multiple languages without requiring special knowledge for each of the languages.

From a technical perspective, we have addressed the problem of using learned classifiers in a pipeline fashion, where a task is decomposed into several stages and classifiers are used sequentially to solve each stage. This is a common computational strategy in natural language processing and is known to suffer from error accumulation and an inability to correct mistakes in previous stages. We abstracted two natural principles, one which calls for making the local classifiers used in the computation more reliable and a second, which suggests to devise the pipeline algorithm in such a way that it minimizes the number of actions taken.

However, since we tried to build a single approach for all languages, we have not fully utilized the capabilities of our algorithms. In future work we will try to specify both features and local search parameters to the target language.

Acknowledgement This research is supported by NSF ITR IIS-0428472, a DOI grant under the Reflex program and ARDA's Advanced Question Answering for Intelligence (AQUAINT) program.

References

A. V. Aho, R. Sethi, and J. D. Ullman. 1986. Compilers: Principles, techniques, and tools. In *Addison-Wesley Publishing Company, Reading, MA*.

A. Carlson, C. Cumby, J. Rosen, and D. Roth. 1999. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department, May.

M. Chang, Q. Do, and D. Roth. 2006. Local search for bottom-up dependency parsing. Technical report, UIUC Computer Science Department.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 806–813.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *IWPT2003*.

Multi-lingual Dependency Parsing at NAIST

Yuchang CHENG, Masayuki ASAHARA and Yuji MATSUMOTO
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
{yuchan-c, masayu-a, matsu}@is.naist.jp

Abstract

In this paper, we present a framework for multi-lingual dependency parsing. Our bottom-up deterministic parser adopts Nivre's algorithm (Nivre, 2004) with a preprocessor. Support Vector Machines (SVMs) are utilized to determine the word dependency attachments. Then, a maximum entropy method (MaxEnt) is used for determining the label of the dependency relation. To improve the performance of the parser, we construct a tagger based on SVMs to find neighboring attachment as a preprocessor. Experimental evaluation shows that the proposed extension improves the parsing accuracy of our base parser in 9 languages. (Hajič et al., 2004; Simov et al., 2005; Simov and Osenova, 2003; Chen et al., 2003; Böhmová et al., 2003; Kromann, 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit and Martí, 2002; Nilsson et al., 2005; Oflazer et al., 2003; Atalay et al., 2003).

1 Introduction

The presented dependency parser is based on our preceding work (Cheng, 2005a) for Chinese. The parser is a bottom-up deterministic dependency parser based on the algorithm proposed by (Nivre, 2004). A dependency attachment matrix is constructed, in which each element corresponds to a pair of tokens. Each dependency attachment is incrementally constructed, with no crossing constraint. In the parser, SVMs (Vapnik, 1998) deterministically estimate whether a pair of words has either of four relations: right, left, shift and reduce. While dependency attachment is estimated by SVMs, we use a MaxEnt (Ratnaparkhi, 1999) based tagger with the output of the parser to estimate the label of dependency relations. This tagger uses the same features as for the word dependency analysis.

In our preceding work (Cheng, 2005a), we not only adopted the Nivre algorithm with SVMs, but also tried some preprocessing methods. We investigated several preprocessing methods on a Chinese Treebank. In this shared task (Buchholz et. al, 2006), we also investigate which preprocessing method is effective on other languages. We found that only the method that uses a tagger to extract the word dependency attachment between two neighboring words works effectively in most of the languages.

2 System Description

The main part of our dependency parser is based on Nivre's algorithm (Nivre, 2004), in which the dependency relations are constructed by a bottom-up deterministic schema. While Nivre's method uses memory-based learning to estimate the dependency attachment and the label, we use SVMs to estimate the attachment and MaxEnt to estimate

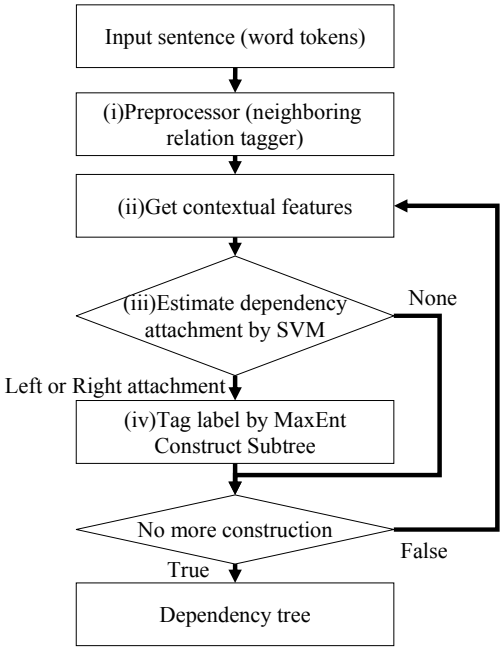


Fig. 1 The architecture of our parser

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X), pages 191–195, New York City, June 2006.

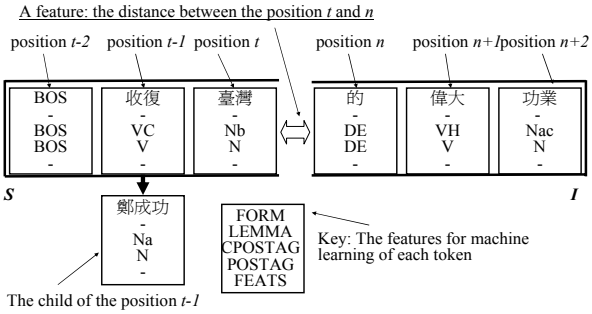


Fig. 2. The features for dependency analysis

the label. The architecture of the parser consists of four major procedures and as in **Fig.1**:

(i) Decide the neighboring dependency attachment between all adjacent words in the input sentence by SVM-based tagger (as a preprocessing)

(ii) Extract the surrounding features for the focused pair of nodes.

(iii) Estimate the dependency attachment operation of the focused pair of nodes by SVMs.

(iv) If there is a left or right attachment, estimate the label of dependency relation by MaxEnt.

We will explain the main procedures (steps (ii)-(iv)) in sections 2.1 and 2.2, and the preprocessing in section 2.3.

2.1 Word dependency analysis

In the algorithm, the state of the parser is represented by a triple $\langle S, I, A \rangle$. *S* and *I* are stacks, *S* keeps the words being in consideration, and *I* keeps the words to be processed. *A* is a list of dependency attachments decided in the algorithm. Given an input word sequence *W*, the parser is initialized by the triple $\langle nil, W, \phi \rangle$. The parser estimates the dependency attachment between two words (the top elements of stacks *S* and *I*). The algorithm iterates until the list *I* becomes empty. There are four possible operations (**Right**, **Left**, **Shift** and **Reduce**) for the configuration at hand.

Right or **Left:** If there is a dependency relation that the word *t* or *n* attaches to word *n* or *t*, add the new dependency relation $(t \rightarrow n)$ or $(n \rightarrow t)$ into *A*, remove *t* or *n* from *S* or *I*.

If there is no dependency relation between *n* and *t,* check the following conditions.

Reduce: If there is no word *n'* ($n' \in I$) which may depend on *t*, and *t* has a parent on its left side, the parser removes *t* from the stack *S*.

Shift: If there is no dependency between *n* and *t,* and the triple does not satisfy the conditions for **Reduce**, then push *n* onto the stack *S*.

In this work, we adopt SVMs for estimating the word dependency attachments. SVMs are binary classifiers based on the maximal margin strategy. We use the polynomial kernel: $K(\mathbf{x},\mathbf{z}) = (1 + \mathbf{x} \cdot \mathbf{z})^d$ with d =2. The performance of SVMs is better than that of the maximum entropy method in our preceding work for Chinese dependency analysis (Cheng, 2005b). This is because that SVMs can combine features automatically (using the polynomial kernel), whereas the maximum entropy method cannot. To extend binary classifiers to multi-class classifiers, we use the pair-wise method, in which we make ${}_nC_2$ [1] binary classifiers between all pairs of the classes (Kreβel, 1998). We use Libsvm (Lin et al., 2001) in our experiments.

In our method, the parser considers the dependency attachment of two nodes (*n,t*). The features of a node are the word itself, the POS-tag and the information of its child node(s). The context features are 2 preceding nodes of node *t* (and *t* itself), 2 succeeding nodes of node *n* (and *n* itself), and their child nodes. The distance between nodes *n* and *t* is also used as a feature. The features are shown in **Fig.2**.

2.2 Label tagging

We adopt MaxEnt to estimate the label of dependency relations. We have tried to use linear-chain conditional random fields (CRFs) for estimating the labels after the dependency relation analysis. This means that the parser first analyzes the word dependency (head-modifier relation) of the input sentence, then the CRFs model analyzes the most suitable label set with the basic information of input sentence (FORM, LEMMA, POSTAG……etc) and the head information (FORM and POSTAG) of each word. However, as the number of possible labels in some languages is large, training a CRF model with these corpora (we use CRF++ (Kudo, 2005)) cost huge memory and time.

Instead, we combine the maximum entropy method in the word dependency analysis to tag the label of dependency relation. As shown in **Fig. 1**, the parser first gets the contextual features to estimate the word dependency. If the parsing operation

[1] To estimate the current operation (Left, Right, Shift and Reduce) by SVMs, we need to build 6 classifiers(Left-Right, Left-Shift, Left-Reduce, Right-Shift, Right-Reduce and Shift-Reduce).

is "Left" or "Right", the parser then use MaxEnt with the same features to tag the label of relation. This strategy can tag the label according to the current states of the focused word pair. We divide the training instances according to the CPOSTAG of the focused word *n*, so that a classifier is constructed for each of distinct POS-tag of the word *n*.

2.3 Preprocessing

2.3.1 Preceding work

In our preceding work (Cheng, 2005a), we discussed three problems of our basic methods (adopt Nivre's algorithm with SVMs) and proposed three preprocessing methods to resolve these problems. The methods include: (1) using global features and a two-steps process to resolve the ambiguity between the parsing operations "Shift" and "Reduce". (2) using a root node finder and dividing the sentence at the root node to make use of the top-down information. (3) extracting the prepositional phrase (PP) to resolve the problem of identifying the boundary of PP.

We incorporated Nivre's method with these preprocessing methods for Chinese dependency analysis with Penn Chinese Treebank and Sinica Treebank (Chen et al., 2003). This was effective because of the properties of Chinese: First, there is no multi-root in Chinese Treebank. Second, the boundary of prepositional phrases is ambiguous. We found that these methods do not always improve the accuracy of all the languages in the shared task.

We have tried the method (1) in some languages to see if there is any improvement in the parser. We attempted to use global features and two-step analysis to resolve the ambiguity of the operations. In Chinese (Chen et al., 2003) and Danish (Kromann, 2003), this method can improve the parser performance. However, in other languages, such as Arabic (Hajič et al., 2004), this method decreased the performance. The reason is that the sentence in some languages is too long to use global features. In our preceding work, the global features include the information of all the un-analyzed words. However, for analyzing long sentences, the global features usually include some useless information and will confuse the two-step process. Therefore, we do not use this method in this shared task.

In the method (2), we construct an SVM-based root node finder to identify the root node and divided the sentence at the root node in the Chinese Treebank. This method is based on the properties of dependency structures "One and only one element is independent" and "An element cannot have modifiers lying on the other side of its own head". However, there are some languages that include multi-root sentences, such as Arabic, Czech, and Spanish (Civit and Martí, 2002), and it is difficult to divide the sentence at the roots. In multi-root sentences, deciding the head of the words between roots is difficult. Therefore, we do not use the method (2) in the share task.

The method (3) –namely PP chunker– can identify the boundary of PP in Chinese and resolve the ambiguity of PP boundary, but we cannot guarantee that to identify the boundary of PP can improve the parser in other languages. Even we do not understand construction of PP in all languages. Therefore, for the robustness in analyzing different languages, we do not use this method.

2.3.2 Neighboring dependency attachment tagger

In the bottom-up dependency parsing approach, the features and the strategies for parsing in early stage (the dependency between adjacent[2] words) is different from parsing in upper stage (the dependency between phrases). Parsing in upper stage needs the information at the phrases not at the words alone. The features and the strategies for parsing in early and upper stages should be separated into distinct. Therefore, we divide the neighboring dependency attachment (for early stage) and normal dependency attachment (for upper stage), and set the neighboring dependency attachment tagger as a preprocessor.

When the parser analyzes an input sentence, it extracts the neighboring dependency attachments first, then analyzes the sentence as described before. The results show that tagging the neighboring dependency word-pairs can improve 9 languages out of 12 scoring languages, although in some languages it degrades the performance a little. Potentially, there may be a number of ways for decomposing the parsing process, and the current method is just the simplest decomposition of the process. The best method of decomposition or dynamic changing of parsing models should be investigated as the future research.

[2] We extract all words that depend on the adjacent word (right or left).

3 Experiment

3.1 Experimental setting

Our system consists of three parts; first, the SVM-based tagger extracts the neighboring attachment relations of the input sentence. Second, the parser analyzes further dependency attachments. If a new dependency attachment is generated, the MaxEnt based tagger estimates the label of the relation. The three parts of our parser are trained on the available data of the languages.

In our experiment, we used the full information of each token (FORM, LEMMA, CPOSTAG, POSTAG, FEATS) when we train and test the model. **Fig. 2** describes the features of each token. Some languages do not include all columns; such that the Chinese data does not include LEMMA and FEATURES, these empty columns are shown by the symbol "-" in **Fig. 2**. The features for the neighboring dependency tagging are the information of the focused word, two preceding words and two succeeding words. **Fig. 2** shows the window size of our features for estimating the word dependency in the main procedures. These features include the focused words (*n*, *t*), two preceding words and two succeeding words and their children. The features for estimating the relation label are the same as the features used for word dependency analysis. For example, if the machine learner estimates the operation of this situation as "**Left**" or "**Right**" by using the features in **Fig. 2**, the parser uses the same features in **Fig. 2** and the dependency relation to estimate the label of this relation.

For training the models efficiently, we divided the training instances of all languages at the CPOSTAG of the focused word *n* in **Fig .2.** In our preceding work, we found this procedure can get better performance than training with all the instances at once. However, only the instances in Czech are divided at the CPOSTAG of the focused word-pair *t*-*n*[3]. The performance of this procedure is worse than using the CPOSTAG of the focused word *n*, because the training instances of each CPOSTAG-pair will become scarce. However, the data size of Czech is much larger than other languages; we couldn't finish the training of Czech using the CPOSTAG of the focused word *n*, before the deadline for submitting. Therefore we used this procedure only for the experiment of Czech.

[3] For example, we have 15 SVM-models for Arabic according to the CPOSTAG of Arabic (A, C, D, F, G…etc.). However, we have 139 SVM-models for Czech according the CPOSTAG pair of focused words (A-A, A-C, A-D…etc.)

All our experiments were run on a Linux machine with XEON 2.4GHz and 4.0GB memory. The program is implemented in JAVA.

3.2 Results

Table 1 shows the results of our parser. We do not take into consideration the problem of cross relation. Although these cross relations are few in training data, they would make our performance worse in some languages. We expect that this is one reason that the result of Dutch is not good. The average length of sentences and the size of training data may have affected the performance of our parser. Sentences of Arabic are longer and training data size of Arabic is smaller than other languages; therefore our parser is worse in Arabic. Similarly, our result in Turkish is also not good because the data size is small.

We compare the result of Chinese with our preceding work. The score of this shared task is better than our preceding work. It is expected that we selected the FORM and CPOSTAG of each nodes as features in the preceding work. However, the POSTAG is also a useful feature for Chinese, and we grouped the original POS tags of Sinica Treebank from 303 to 54 in our preceding work. The number of CPOSTAG(54) in our preceding work is more than the number of CPOSTAG(22) in this shared task, the training data of each CPOSTAG in our preceding work is smaller than in this work. Therefore the performance of our preceding work in Sinica Treebank is worse than this task.

The last column of the **Table 1** shows the unlabeled scores of our parser without the preprocessing. Because our parser estimates the label after the dependency relation is generated. We only consider whether the preprocessing can improve the unlabeled scores. Although the preprocessing can not improve some languages (such as Chinese, Spanish and Swedish), the average score shows that using preprocessing is better than parsing without preprocessing.

Comparing the gold standard data and the system output of Chinese, we find the CPOSTAG with lowest accuracy is "P (preposition)", the accuracy that both dependency and head are correct is 71%. As we described in our preceding work and Section 2.3, we found that boundaries of prepositional phrases are ambiguous for Chinese. The bottom-up algorithm usually wrongly parses the prepositional phrase short. The parser does not capture the correct information of the children of the preposition. According to the results, this problem does not cause the accuracy of head of

Language:	LAS:	UAS:	LAcc.	UAS with out preprocessing:
Arabic	65.19	77.74	79.02	76.74
Chinese	84.27	89.46	86.42	90.03
Czech	76.24	83.4	83.52	82.88
Danish	81.72	88.64	86.11	88.45
Dutch	71.77	75.49	75.83	74.97
German	84.11	87.66	90.67	87.53
Japanese	89.91	93.12	92.40	92.99
Portugese	85.07	90.3	88.00	90.21
Slovene	71.42	81.14	80.96	80.43
Spanish	80.46	85.15	88.90	85.19
Swedish	81.08	88.57	83.99	88.83
Turkish	61.22	74.49	73.91	74.3
AV:	77.7	84.6	84.1	84.38
SD:	8.67	6.15	5.78	6.42
Bulgarian	86.34	91.3	89.27	91.44

Table 1: Results

CPOSTAG "P" decrease. Actually, the head accuracy of "P" is better than the CPOSTAG "C" or "V". However, the dep. accuracy of "P" is worse. We should consider the properties of prepositions in Chinese to resolve this question. In Chinese, prepositions are derived from verbs; therefore some prepositions can be used as a verb. Naturally, the dependency relation of a preposition is different from that of a verb. Important information for distinguishing whether the preposition is a verb or a preposition is the information of the children of the preposition. The real POS tag of a preposition which includes few children is usually a verb; on the other hand, the real POS tag of a preposition is usually a preposition.

If our parser considers the preposition which leads a short phrase, the parser will estimate the relation of the preposition as a verb. At the same time, if the boundary of prepositional phrase is analyzed incorrectly, other succeeding words will be wrongly analyzed, too.

Error analysis of Japanese data (Kawata and Bartels, 2000) shows that CNJ (Conjunction) is a difficult POS tag. The parser does not have any module to detect coordinate structures. (Kurohashi, 1995) proposed a method in which coordinate structure with punctuation is detected by a coefficient of similarity. Similar framework is necessary for solving the problem.

Another characteristic error in Japanese is seen at adnominal dependency attachment for a compound noun. In such dependency relations, adjectives and nouns with "no" (genitive marker) can be a dependent and compound nouns which consist of more than one consecutive nouns can be a head. The constituent of compound nouns have same POSTAG, CPOSTAG and FEATS. So, the machine learner has to disambiguate the dependency attachment with sparce feature LEMMA and FORM. Compound noun analysis by semantic feature is necessary for addressing the issue.

4 Conclusion

This paper reported on multi-lingual dependency parsing on combining SVMs and MaxEnt. The system uses SVMs for word dependency attachment analysis and MaxEnt for the label tagging when the new dependency attachment is generated. We discussed some preprocessing methods that are useful in our preceding work for Chinese dependency analysis, but these methods, except one, cannot be used in multi-lingual dependency parsing. Only using the SVM-based tagger to extract the neighbor relation could improve many languages in our experiment, therefore we use the tagger in the parser as its preprocessing.

References

S. Buchholz, E. Marsi, A. Dubey and Y. Krymolowski. 2006. *CoNLL-X: Shared Task on Multilingual Dependency Parsing*, CoNLL 2006.

Yuchang Cheng, Masayuki Asahara and Yuji Matsumoto. 2005a. *Chinese Deterministic Dependency Parser: Examining Effects of Global Features and Root Node Finder*, Fourth SIGHAN Workshop, pp.17-24.

Yuchang Cheng, Masayuki Asahara and Yuji Matsumoto. 2005b. *Machine Learning-based Dependency Parser for Chinese*, the International Conference on Chinese Computing, pp.66-73.

Ulrich. H.-G. Kreβel, 1998. *Pairwise classification and support vector machines*. In Advances in Kernel Methods, pp. 255-268. The MIT Press.

Taku Kudo. *CRF++: Yet Another CRF toolkit*, http://www.chasen.org/~taku/software/CRF++/.

Sadao Kurohashi. 1995. *Analyzing Coordinate Structures Including Punctuation in English*, In IWPT-95, pp. 136-147.

Chih Jen Lin, 2001. *A practical guide to support vector classification*, http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Joakim Nivre, 2004. *Incrementality in Deterministic Dependency Parsing*, In Incremental Parsing: Bringing Engineering and Cognition Together. Workshop at ACL-2004, pp. 50-57.

Adwait Ratnaparkhi, 1999. *Learning to parse natural language with maximum entropy models.* Machine Learning, 34(1-3):151-175.

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

Dependency Parsing with Reference to Slovene, Spanish and Swedish

Simon Corston-Oliver
Natural Language Processing
Microsoft Research
One Microsoft Way
Redmond WA 98052
simonco@microsoft.com

Anthony Aue
Natural Language Processing
Microsoft Research
One Microsoft Way
Redmond WA 98052
anthaue@microsoft.com

Abstract

We describe a parser used in the CoNLL 2006 Shared Task, "Multingual Dependency Parsing." The parser first identifies syntactic dependencies and then labels those dependencies using a maximum entropy classifier. We consider the impact of feature engineering and the choice of machine learning algorithm, with particular focus on Slovene, Spanish and Swedish.

1 Introduction

The system that we submitted for the CoNLL 2006 Shared Task, "Multingual Dependency Parsing," (Buchholz et al., 2006) is a two stage pipeline. The first stage identifies unlabeled directed dependencies using an extension of the parser described in (Corston-Oliver et al., 2006). The second stage is a maximum entropy classifier that labels the directed dependencies. The system was trained on the twelve obligatory languages, as well as the optional language, Bulgarian (Hajič et al., 2004; Simov et al., 2005; Simov and Osenova, 2003; Chen et al., 2003; Böhmová et al., 2003; Kromann, 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit Torruella and Martí Antonín, 2002; Nilsson et al., 2005; Oflazer et al., 2003; Atalay et al., 2003).

Table 1 presents the results of the system described in the current paper on the CoNLL shared task, including the optional evaluation on Bulgarian. For Slovene, we ranked second with a labeled dependency accuracy of 72.42%. This was not statistically significantly different from the top-ranked score of 73.44%. For Spanish, our labeled dependency accuracy of 80.36% is within 0.1% of the third-ranked score of 80.46%. Our unlabeled dependency accuracy for Swedish was the best of all the systems at 89.54%. Our labeled accuracy for Swedish, however, at 79.69%, fell far short of the third-best score of 82.31%. We therefore focus on Swedish when considering the impact of our choice of learning algorithm on our label accuracy.

Language	Unlabeled Attachment	Labeled Attachment
Arabic	78.40	63.53
Bulgarian	90.09	83.36
Chinese	90.00	79.92
Czech	83.02	74.48
Danish	87.94	81.74
Dutch	74.83	71.43
German	87.20	83.47
Japanese	92.84	89.95
Portugese	88.96	84.59
Slovene	81.77	72.42
Spanish	84.87	80.36
Swedish	89.54	79.69
Turkish	73.11	61.74

Table 1: Results on CoNLL 2006 shared task.

2 Data

We divided the shared data into training and development test sets, using larger development test sets

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 196–200, New York City, June 2006.

for the languages supplied with more data. The development test set consisted of 250 sentences for Arabic, Slovene, Spanish and Turkish, 500 sentences for Danish and Portuguese, and 1,000 sentences for the other languages.

3 The Parser

The baseline parser predicts unlabeled directed dependencies. As described in (Corston-Oliver et al., 2006), we reimplemented the parser described in (McDonald et al., 2005) and validated their results for Czech and English.

The parser finds the highest-scoring parse $\hat{y}$ among all possible parses $y \in Y$ for a given sentence:

$$\hat{y} = \arg\max_{y \in Y} s(y) \quad (1)$$

The score s of a given parse y is the sum of the scores of all the dependency links (i,j) $\in y$:

$$s(y) = \sum_{(i,j) \in y} d(i,j) = \sum_{(i,j) \in y} \mathbf{w} \cdot \mathbf{f}(i,j) \quad (2)$$

where the link (i,j) indicates a parent-child dependency between the token at position i and the token at position j. The score $d(i,j)$ of each dependency link (i,j) is further decomposed as the weighted sum of its features $\mathbf{f}(i,j)$.

To set **w**, we trained twenty averaged perceptrons on different shuffles of the training data, using the development test set to determine when the perceptrons had converged. The averaged perceptrons were then combined to make a Bayes Point Machine (Harrington et al., 2003). At both training and run time, edges are scored independently, and Eisner's $O(N^3)$ decoder (Eisner, 1996) is used to find the optimal parse. This decoder produces only projective analyses, although it does allow for analyses with multiple roots.

The features used for scoring the edges prior to applying Eisner's algorithm are extracted from each possible parent-child dependency. The features include the case-normalized original form and lemma[1] of each token , the part of speech (POS) tag of each token, the POS tag of each intervening token and of each token to the left and right of the parent and child. Additional features are created by combining these atomic features, as described in (McDonald et al., 2005). All features are in turn combined with the direction of attachment and the distance between tokens. Distance was discretized, with individual buckets for distances 0-4, a single bucket for 5-9, and a single bucket for 10+. In sections 3.1 and 3.2 we discuss the feature engineering we performed.

[1]If no lemma was specified, we truncated the original form by taking the first two characters for Chinese words consisting of two characters or more and the first five characters for words consisting of five characters or more in the other languages.

3.1 Part of Speech Features

We experimented with using the coarse POS tag and the fine POS tag. In our official submission, we used fine POS tags for all languages except Dutch and Turkish. For Dutch and Turkish, using the fine POS tag resulted in a reduction in unlabeled dependency accuracy of 0.12% and 0.43% respectively on the development test sets, apparently because of the sparsity of the fine POS tags. For German and Swedish, the fine and coarse POS tags are the same so using the fine POS tag had no effect. For other languages, using the fine POS tag showed modest improvements in unlabeled dependency accuracy.

For Swedish, we performed an additional manipulation on the POS tags, normalizing the distinct POS tags assigned to each verbal auxiliary and modal to a single tag "aux". For example, in the Swedish data all inflected forms of the verb "vara" ("be") are tagged as AV, and all inflected forms of the modal "måste" ("must") are tagged as MV. This normalization caused unlabeled dependency accuracy on the Swedish development set to improve from 89.23% to 89.45%.

3.2 Features for Root Identification

Analysis of the baseline parser's errors suggested the need for additional feature types to improve the identification of the root of the sentence. In particular, the parser was frequently making errors in identifying the root of periphrastic constructions involving an auxiliary verb or modal and a participle. In Germanic languages, for example, the auxiliary or modal typically occurs in second position in declarative main clauses or in initial position in cases of subject-aux inversion. We added a collection of features intended to improve the identification of the root. The hope was that improved root identification would have a positive cascading effect in the

identification of other dependencies, since a failure to correctly identify the root of the sentence usually means that the parse will have many other errors.

We extracted four feature types, the original form of the first and last tokens in the sentence and the POS of the first and last tokens in the sentence. These features were intended to identify declarative vs. interrogative sentences.

For each child and parent token being scored, we also noted the following four features: "child/parent is first non-punctuation token in sentence", "child/parent is second non-punctuation token in sentence". The features that identify the second token in the sentence were intended to improve the identification of verb-second phenomena. Of course, this is a linguistic oversimplification. Verb-second phenomena are actually sensitive to the order of constituents, not words. We therefore added four feature types that considered the sequence of POS tags to the left of the child or parent if they occurred within ten tokens of the beginning of the sentence and the sequence of POS tags to the right of the child or parent if they occurred within ten tokens of the end of the sentence.

We also added features intended to improve the identification of the root in sentences without a finite verb. For example, the Dutch training data contained many simple responses to a question-answering task, consisting of a single noun phrase. Four simple features were used "Child/Parent is the leftmost noun in the sentence", "Child/Parent is a noun but not the leftmost noun in the sentence". These features were combined with an indicator "Sentence contains/does not contain a finite verb".

Child or parent tokens that were finite verbs were flagged as likely candidates for being the root of the sentence if they were the leftmost finite verb in the sentence and not preceded by a subordinating conjunction or relative pronoun. Finite verbs were identified by POS tags and morphological features, e.g. in Spanish, verbs without the morphological feature "mod=n" were identified as finite, while in Portuguese the fine POS tag "v-fin" was used.

Similarly, various sets of POS tags were used to identify subordinating conjunctions or relative pronouns for different languages. For example, in Bulgarian the fine POS tag "pr" (relative pronoun) and "cs" (subordinating conjunction) were used. For Dutch, the morphological features "onder", "betr" and "voorinf" were used to identify subordinating conjunctions and relative pronouns.

These features wreaked havoc with Turkish, a verb-final language. For certain other languages, dependency accuracy measured on the development test set improved by a modest amount, with more dramatic improvements in root accuracy (F1 measure combining precision and recall for non-punctuation root tokens).

Since the addition of these features had been motivated by verb-second phenomena in Germanic languages, we were surprised to discover that the only Germanic language to demonstrate a marked improvement in unlabeled dependency accuracy was Danish, whose accuracy on the development set rose from 87.51% to 87.72%, while root accuracy F1 rose from 94.12% to 94.72%. Spanish showed a modest improvement in unlabeled dependency accuracy, from 85.08% to 85.13%, but root F1 rose from 80.08% to 83.57%.

The features described above for identifying the leftmost finite verb not preceded by a subordinating conjunction or relative pronoun did not improve Slovene unlabeled dependency accuracy, and so were not included in the set of root-identifying features in our Slovene CoNLL submission. Closer examination of the Slovene corpus revealed that periphrastic constructions consisting of one or more auxiliaries followed by a participle were annotated with the participle as the head, whereas for other languages in the shared task the consensus view appears to be that the auxiliary should be annotated as the head. Singling out the leftmost finite verb in Slovene when a participle ought to be selected as the root of the sentence is therefore counter-productive. The other root identification features did improve root F1 in Slovene. Root F1 on the development test set rose from 45.82% to 46.43%, although overall unlabeled dependency accuracy on the development test set fell slightly from 80.24% to 79.94%.

3.3 Morphological Features

As the preceding discussion shows, morphological information was occasionally used to assist in making finer-grained POS distinctions than were made in the POS tags, e.g., for distinguishing subordinating vs. coordinating conjunctions. Aside from

these surgical uses of the morphological information present in the CoNLL data, morphology was not explicitly used by the baseline parser. For example, there were no features that considered subject-verb agreement nor agreement of an adjective with the number or lexical gender of the noun it modified. However, it is possible that morphological information influenced the training of edge weights if the information was implicit in the POS tags.

4 The Dependency Labeler

4.1 Classifier

We used a maximum entropy classifier (Berger et al., 1996) to assign labels to the unlabeled dependencies produced by the Bayes Point Machine. We used the same training and development test split that was used to train the dependency parser. We chose to use maximum entropy classifiers because they can be trained relatively quickly while still offering reasonable classification accuracy and are robust in the face of large numbers of superfluous features, a desirable property given the requirement that the same parser handle multiple languages. Furthermore, maximum entropy classifiers provide good probability distributions over class labels. This was important to us because we had initially hoped to find the optimal set of dependency labels for the children of a given node by modeling the probability of each set of labels conditioned on the lemma and POS of the parent. For example, labeling each dependant of a parent node independently might result in three OBJECT relations dependent on a single verb; modeling sets of relations ought to prevent this. Unfortunately, this approach did not outperform labeling each node independently.

Therefore, the system we submitted labeled each dependency independently, using the most probable label from the maximum entropy classifier. We have noted in previous experiments that our SVM implementation often gives better one-best classification accuracy than our maximum entropy implementation, but did not have time to train SVM classifiers.

To see how much the choice of classification algorithm affected our official results, we trained a linear SVM classifier for Swedish after the competition had ended, tuning parameters on the development test set. As noted in section 1, our system scored highest for Swedish in unlabeled dependency accuracy at 89.54% but fell well short of the third-ranked system when measuring labeled dependency accuracy. Using an SVM classifier instead of a maximum entropy classifier, Swedish label accuracy rose from 82.33% to 86.06%, and labeled attachment accuracy rose from 79.69% to 82.95%, which falls between the first-ranked score of 84.58% and the second-ranked score of 82.55%. Similarly, Japanese label accuracy rose from 93.20% to 93.96%, and labeled attachment accuracy rose from 89.95% to 90.77% when we replaced a maximum entropy classifier with an SVM. This labeled attachment result of 90.77% is comparable to the official second place result of 90.71% for Japanese. We conclude that a two stage pipeline such as ours, in which the second stage labels dependencies in isolation, is greatly impacted by the choice of classifier.

4.2 Features Used for Labeling

We extracted features from individual nodes in the dependency tree, parent-child features and features that took nodes other than the parent and child into account.

The features extracted from each individual parent and child node were the original surface form, the lemma (see footnote 1 above), the coarse and fine POS tags and each morphological feature.

The parent-child features are the direction of modification, the combination of the parent and child lemmata, all combinations of parent and child lemma and coarse POS tag (e.g. child lemma combined with coarse POS tag of the parent) and all pairwise combinations of parent and child morphology features (e.g. parent is feminine and child is plural).

Additional features were verb position (whether the parent or child is the first or last verb in the sentence), coarse POS and lemma of the left and right neighbors of the parent and child, coarse POS and lemma of the grandparent, number and coarse POS tag sequence of siblings to the left and to the right of the child, total number of siblings of the child, number of tokens governed by child, whether the parent has a verbal ancestor, lemma and morphological features of the verb governing the child (if any), and coarse POS tag combined with relative offset of each sibling (e.g., the sibling two to the left of the child is a determiner).

For Slovene, the label accuracy using all of the features above was 81.91%. We retrained our maximum entropy classifier by removing certain classes of features in order to determine their contribution. Removing the weight features caused a notable drop, with label accuracy on the development test set falling 0.52% to 81.39%. Removing the grandparent features (but including weight features) caused an even greater drop of 1.03% to 80.88%. One place where the grandparent features were important was in distinguishing between Adv and Atr relations. It appears that the relation between a noun and its governing preposition or between a verb and its governing conjunction is sensitive to the part of speech of the grandparent. For example, we observed a number of cases where the relation between a noun and its governing preposition had been incorrectly labeled as Adv when it should have been Atr. The addition of grandparent features allowed the classifier to make the distinction by looking at the POS of the grandparent; when the POS was noun, the classifier tended to correctly choose the Atr label.

5 Conclusion

We have described a two stage pipeline that first predicts directed unlabeled dependencies and then labels them. The system performed well on Slovene, Spanish and Swedish. Feature engineering played an important role both in predicting dependencies and in labeling them. Finally, replacing the maximum entropy classifier used to label dependencies with an SVM improves upon our official results.

References

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

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of the Tenth Conf. on Computational Natural Language Learning (CoNLL-X)*. SIGNLL.

Simon Corston-Oliver, Anthony Aue, Kevin Duh, and Eric Ringger. 2006. Multilingual dependency parsing using bayes point machines. In *Proc. of HLT-NAACL 2006*.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of the 16th Intern. Conf. on Computational Linguistics (COLING)*, pages 340–345.

Edward Harrington, Ralf Herbrich, Jyrki Kivinen, John C. Platt, and Robert C. Williamson. 2003. Online bayes point machines. In *Proceedings of Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 241–252.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Assocation for Computational Linguistics*.

Vine Parsing and Minimum Risk Reranking for Speed and Precision*

Markus Dreyer, David A. Smith, and **Noah A. Smith**
Department of Computer Science / Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218 USA
{markus,{d,n}asmith}@cs.jhu.edu

Abstract

We describe our entry in the CoNLL-X shared task. The system consists of three phases: a probabilistic vine parser (Eisner and N. Smith, 2005) that produces unlabeled dependency trees, a probabilistic relation-labeling model, and a discriminative minimum risk reranker (D. Smith and Eisner, 2006). The system is designed for fast training and decoding and for high precision. We describe sources of cross-lingual error and ways to ameliorate them. We then provide a detailed error analysis of parses produced for sentences in German (much training data) and Arabic (little training data).

1 Introduction

Standard state-of-the-art parsing systems (e.g., Charniak and Johnson, 2005) typically involve two passes. First, a **parser** produces a list of the most likely n parse trees under a generative, probabilistic model (usually some flavor of PCFG). A discriminative **reranker** then chooses among trees in this list by using an extended feature set (Collins, 2000). This paradigm has many advantages: PCFGs are fast to train, can be very robust, and perform better as more data is made available; and rerankers train quickly (compared to discriminative models), require few parameters, and permit arbitrary features.

We describe such a system for **dependency** parsing. Our shared task entry is a preliminary system developed in only 3 person-weeks, and its accuracy is typically one s.d. below the average across systems and 10–20 points below the best system. On the positive side, its decoding algorithms have guaranteed $O(n)$ runtime, and training takes only a couple of hours. Having designed primarily for **speed** and **robustness**, we sacrifice accuracy. Better **estimation**, reranking on larger datasets, and more fine-grained parsing constraints are expected to boost accuracy while maintaining speed.

2 Notation

Let a sentence $\mathbf{x} = \langle x_1, x_2, ..., x_n \rangle$, where each x_i is a tuple containing a part-of-speech tag t_i and a word w_i, and possibly more information.[1] x_0 is a special wall symbol, \$, on the left. A dependency tree $\mathbf{y}$ is defined by three functions: $\mathbf{y}_{left}$ and $\mathbf{y}_{right}$ (both $\{0, 1, 2, ..., n\} \rightarrow 2^{\{1,2,...,n\}}$) that map each word to its sets of left and right dependents, respectively, and $\mathbf{y}_{label} : \{1, 2, ..., n\} \rightarrow \mathcal{D}$, which labels the relationship between word i and its parent from label set $\mathcal{D}$.

In this work, the graph is constrained to be a *projective* tree rooted at \$: each word except \$ has a single parent, and there are no cycles or crossing dependencies. Using a simple dynamic program to find the minimum-error projective parse, we find that assuming projectivity need not harm accuracy very much (Tab. 1, col. 3).

3 Unlabeled Parsing

The first component of our system is an unlabeled parser that, given a sentence, finds the U best unlabeled trees under a probabilistic model using a bottom-up dynamic programming algorithm.[2] The model is a probabilistic head automaton grammar (Alshawi, 1996) that assumes conditional indepen-

*This work was supported by NSF ITR grant IIS-0313193, an NSF fellowship to the second author, and a Fannie and John Hertz Foundation fellowship to the third author. The views expressed are not necessarily endorsed by the sponsors. We thank Charles Schafer, Keith Hall, Jason Eisner, and Sanjeev Khudanpur for helpful conversations.

[1] We used words and fine tags in our parser and labeler, with coarse tags in one backoff model. Other features are used in reranking; we never used the given morphological features or the "projective" annotations offered in the training data.

[2] The execution model we use is best-first, exhaustive search, as described in Eisner et al. (2004). All of our dynamic programming algorithms are implemented concisely in the Dyna language.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 201–205, New York City, June 2006.

	B_ℓ	B_r	projective oracle	(B_ℓ, B_r)-vine oracle	20-best unlabeled oracle	1-best unlabeled	20×50-best labeled oracle, reranked	1×1-best labeled	**reranked** (labeled)	(unlabeled)	(non-$ unl. recall)	(non-$ unl. precision)	
Arabic	10	4	**99.8**	**90.7**	**71.5**	68.1	68.7	59.7	52.0	53.4	68.5	63.4	76.0
Bulgarian	5	4	**99.6**	**90.7**	86.4	80.1	80.5	85.1	73.0	74.8	82.0	74.3	86.3
Chinese	4	4	**100.0**	**93.1**	89.9	79.4	*77.7*	88.6	72.6	*71.6*	77.6	61.4	80.8
Czech	6	4	**97.8**	**90.5**	79.2	70.3	71.5	72.8	58.1	60.5	70.7	64.8	75.7
Danish	5	4	**99.2**	**91.4**	**84.6**	77.7	78.6	79.3	65.5	66.6	77.5	71.4	83.4
Dutch	6	5	**94.6**	**88.3**	77.5	67.9	68.8	73.6	59.4	61.6	68.3	60.4	73.0
German	8	7	**98.8**	**90.9**	83.4	75.5	76.2	82.3	70.1	71.0	77.0	70.2	82.9
Japanese	4	1	**99.2**	**92.2**	90.7	86.3	*85.1*	89.4	81.6	82.9	86.0	68.5	91.5
Portuguese	5	5	**98.8**	**91.5**	85.9	81.4	82.5	83.7	73.4	75.3	82.4	76.2	87.0
Slovene	6	4	**98.5**	**91.7**	**80.5**	72.0	73.3	72.8	57.5	58.7	72.9	66.3	78.5
Spanish	5	6	**100.0**	**91.2**	77.3	71.5	72.6	74.9	66.2	67.6	72.9	69.3	80.7
Swedish	4	5	**99.7**	**94.0**	**87.5**	79.3	79.6	81.0	65.5	67.6	79.5	72.6	83.3
Turkish	6	1	**98.6**	**89.5**	**73.0**	61.0	61.8	64.4	44.9	46.1	60.5	48.5	61.6
						parser	reranker	labeler			reranker		
	1	2	3	4	5	6	7	8	9	10	11	12	13

Table 1: Parameters and performance on test data. B_ℓ and B_r were chosen to retain 90% of dependencies in training data. We show oracle, 1-best, and reranked performance on the test set at different stages of the system. Boldface marks oracle performance that, given perfect downstream modules, would supercede the best system. Italics mark the few cases where the reranker increased error rate. Columns 8–10 show labeled accuracy; column 10 gives the final shared task evaluation scores.

dence between the left yield and the right yield of a given head, given the head (Eisner, 1997).[3] The best known parsing algorithm for such a model is $O(n^3)$ (Eisner and Satta, 1999). The U-best list is generated using Algorithm 3 of Huang and Chiang (2005).

3.1 Vine parsing (dependency length bounds)

Following Eisner and N. Smith (2005), we also impose a bound on the string distance between every child and its parent, with the exception of nodes attaching to \$. Bounds of this kind are intended to improve precision of non-\$ attachments, perhaps sacrificing recall. Fixing bound B_ℓ, no left dependency may exist between child x_i and parent x_j such that $j - i > B_\ell$ (similarly for right dependencies and B_r). As a result, edge-factored parsing runtime is reduced from $O(n^3)$ to $O(n(B_\ell^2 + B_r^2))$. For each language, we choose B_ℓ (B_r) to be the minimum value that will allow recovery of 90% of the left (right) dependencies in the training corpus (Tab. 1, cols. 1, 2, and 4). In order to match the training data to the parsing model, we re-attach disallowed long dependencies to \$ during training.

3.2 Estimation

The probability model predicts, for each parent word x_j, $\{x_i\}_{i \in \mathbf{y}_{left}(j)}$ and $\{x_i\}_{i \in \mathbf{y}_{right}(j)}$. An advantage of head automaton grammars is that, for a given parent node x_j, the children on the same side, $\mathbf{y}_{left}(j)$,

[3] To empirically test this assumption across languages, we measured the mutual information between different features of $\mathbf{y}_{left}(j)$ and $\mathbf{y}_{right}(j)$, given x_j. (Mutual information is a statistic that equals zero iff conditional independence holds.) A detailed discussion, while interesting, is omitted for space, but we highlight some of our findings. First, unsurprisingly, the split-head assumption appears to be less valid for languages with freer word order (Czech, Slovene, German) and more valid for more fixed-order languages (Chinese, Turkish, Arabic) or corpora (Japanese). The children of verbs and conjunctions are the most frequent violators. The mutual information between the sequence of dependency labels on the left and on the right, given the head's (coarse) tag, only once exceeded 1 bit (Slovene).

for example, can depend on each other (cf. McDonald et al., 2005). Child nodes in our model are generated outward, conditional on the parent and the most recent same-side sibling (MRSSS). This increases our parser's theoretical runtime to $O(n(B_\ell^3 + B_r^3))$, which we found was quite manageable.

Let $\mathrm{par}_{\mathbf{y}} : \{1, 2, ..., n\} \rightarrow \{0, 1, ..., n\}$ map each node to its parent in $\mathbf{y}$. Let $\mathrm{pred}_{\mathbf{y}} : \{1, 2, ..., n\} \rightarrow \{\emptyset, 1, 2, ..., n\}$ map each node to the MRSSS in $\mathbf{y}$ if it exists and $\emptyset$ otherwise. Let $\Delta_i = |i - j|$ if j is i's parent. Our (probability-deficient) model defines

$$p(\mathbf{y}) = \prod_{j=1}^{n} \left(\prod_{i \in \mathbf{y}_{left}(j)} p(x_i, \Delta_i \mid x_j, x_{\mathrm{pred}_{\mathbf{y}}(i)}, \textit{left}) \right) \times p(\mathrm{STOP} \mid x_j, x_{\min_{\mathbf{y}_{left}(j)} j}, \textit{left}) \times \left(\prod_{i \in \mathbf{y}_{right}(j)} p(x_i, \Delta_i \mid x_j, \mathrm{pred}_{\mathbf{y}}(i), \textit{right}) \right) \times p(\mathrm{STOP} \mid x_j, x_{\max_{\mathbf{y}_{right}(j)} j}, \textit{right}) \quad (1)$$

Due to the familiar sparse data problem, a maximum likelihood estimate for the ps in Eq. 1 performs very badly (2–23% unlabeled accuracy). Good statistical parsers smooth those distributions by making **conditional independence assumptions** among variables, including backoff and factorization. Arguably the choice of assumptions made (or interpolated among) is central to the success of many existing parsers.

Noting that (a) there are exponentially many such options, and (b) the best-performing independence assumptions will almost certainly vary by language, we use a mixture among 8 such models. The same mixture is used for all languages. The models were not chosen with particular care,[4] and the mixture is *not trained*—the coefficients are fixed at uniform, with a unigram coarse-tag model for backoff. In principle, this mixture should be trained (e.g., to maximize likelihood or minimize error on a development dataset).

The performance of our unlabeled model's top choice and the top-20 oracle are shown in Tab. 1, cols. 5–6. In 5 languages (boldface), perfect labeling and reranking at this stage would have resulted in performance superior to the language's best labeled system, although the oracle is never on par with the best *unlabeled* performance.

4 Labeling

The second component of our system is a labeling model that *independently* selects a label from $\mathcal{D}$ for each parent/child pair in a tree. Given the U best unlabeled trees for a sentence, the labeler produces the L best *labeled* trees for each unlabeled one. The computation involves an $O(|\mathcal{D}|n)$ dynamic programming algorithm, the output of which is passed to Huang and Chiang's (2005) algorithm to generate the L-best list.

We separate the labeler from the parser for two reasons: speed and candidate diversity. In principle the vine parser could jointly predict dependency labels along with structures, but parsing runtime would increase by at least a factor of $|\mathcal{D}|$. The two stage process also forces diversity in the candidate list (20 structures with 50 labelings each); the 1,000-best list of *jointly*-decoded parses often contained many (bad) relabelings of the same tree.

In retrospect, assuming independence among dependency labels damages performance substantially for some languages (Turkish, Czech, Swedish, Danish, Slovene, and Arabic); note the often large drop in oracle performance between Tab. 1, cols. 5 and 8. This assumption is necessary in our framework, because the $O(|\mathcal{D}|^{M+1}n)$ runtime of decoding with an Mth-order Markov model of labels[5] is in general prohibitive—in some cases $|\mathcal{D}| > 80$. Pruning and search heuristics might ameliorate runtime.

If x_i is a child of x_j in direction D, and x_{pred} is the MRSSS (possibly $\emptyset$), where $\Delta_i = |i - j|$, we estimate $p(\ell, x_i, x_j, x_{pred}, \Delta_i \mid D)$ by a mixture (untrained, as in the parser) of four backed-off, factored estimates.

After parsing and labeling, we have for each sentence a list of $U \times L$ candidates. Both the oracle performance of the best candidate in the (20×50)-best list and the performance of the top candidate are shown in Tab. 1, cols. 8–9. It should be clear from the drop in both oracle and 1-best accuracy that our labeling model is a major source of error.

[4] Our infrastructure provides a concise, interpreted language for expressing the models to be mixed, so large-scale combination and comparison are possible.

[5] We tested first-order Markov models that conditioned on parent or MRSSS dependency labels.

5 Reranking

We train a log-linear model combining many feature scores (see below), including the log-probabilities from the parser and labeler. Training minimizes the expected error under the model; we use deterministic annealing to smooth the error surface and avoid local minima (Rose, 1998; D. Smith and Eisner, 2006).

We reserved 200 sentences in each language for training the reranker, plus 200 for choosing among rerankers trained on different feature sets and different $(U \times L)$-best lists.[6]

Features Our reranking features predict tags, labels, lemmata, suffixes and other information given all or some of the following non-local conditioning context: bigrams and trigrams of tags or dependency labels; parent and grandparent dependency labels; subcategorization frames (in terms of tags or dependency labels); the occurrence of certain tags between head and child; surface features like the lemma[7] and the 3-character suffix. In some cases the children of a node are considered all together, and in other cases left and right are separated.

The highest-ranked features during training, for all languages, are the parser and labeler probabilities, followed by $p(\Delta_i \mid t_{parent})$, $p(direction \mid t_{parent})$, $p(label \mid label_{pred}, label_{succ}, subcat)$, and $p(\mathrm{coarse}(t) \mid D, \mathrm{coarse}(t_{parent}), \mathrm{Betw})$, where Betw is TRUE iff an instance of the coarse tag type with the highest mutual information between its left and right children (usually verb) is between the child and its head.

Feature and Model Selection For training speed and to avoid overfitting, only a subset of the above features are used in reranking. Subsets of different sizes (10, 20, and 40, plus "all") are identified for each language using two naïve feature-selection heuristics based on independent performance of features. The feature subset with the highest accuracy on the 200 heldout sentences is selected.

[6] In training our system, we made a serious mistake in training the reranker on only 200 sentences. As a result, our pretesting estimates of performance (on data reserved for model selection) were very bad. The reranker, depending on condition, had only 2–20 times as many examples as it had parameters to estimate, with overfitting as the result.

[7] The first 4 characters of a word are used where the lemma is not available.

Performance Accuracy of the top parses after reranking is shown in Tab. 1, cols. 10–11. Reranking almost always gave some improvement over 1-best parsing.[8] Because of the vine assumption and the preprocessing step that re-attaches all distant children to $, our parser learns to over-attach to $, treating $-attachment as a default/agnostic choice. For many applications a local, incomplete parse may be sufficiently useful, so we also measured non-$ unlabeled precision and recall (Tab. 1, cols. 12–13); our parser has $>$ 80% precision on 8 of the languages. We also applied reranking (with unlabeled features) to the 20-best unlabeled parse lists (col. 7).

6 Error Analysis: German

The plurality of errors (38%) in German were erroneous $ attachments. For ROOT dependency labels, we have a high recall (92.7%), but low precision (72.4%), due most likely to the dependency length bounds. Among the most frequent tags, our system has most trouble finding the correct heads of prepositions (APPR), adverbs (ADV), finite auxiliary verbs (VAFIN), and conjunctions (KON), and finding the correct dependency labels for prepositions, nouns, and finite auxiliary verbs.

The German conjunction *und* is the single word with the most frequent head attachment errors. In many of these cases, our system does not learn the subtle difference between enumerations that are headed by *A* in *A und B*, with two children *und* and *B* on the right, and those headed by *B*, with *und* and *A* as children on its left.

Unlike in some languages, our labeled oracle accuracy is nearly as good as our unlabeled oracle accuracy (Tab. 1, cols. 8, 5). Among the ten most frequent dependency labels, our system has the most difficulty with accusative objects (OA), genitive attributes (AG), and postnominal modifiers (MNR). Accusative objects are often mistagged as subject (SB), noun kernel modifiers (NK), or AG. About 32% of the postnominal modifier relations (*ein Platz in der Geschichte*, 'a place in history') are labeled as modifiers (*in die Stadt fliegen*, 'fly into the city'). Genitive attributes are often tagged as NK since both are frequently realized as nouns.

[8] The exception is Chinese, where the training set for reranking is especially small (see fn. 6).

7 Error Analysis: Arabic

As with German, the greatest portion of Arabic errors (40%) involved attachments to $. Prepositions are consistently attached too low and accounted for 26% of errors. For example, if a form in construct (*idafa*) governed both a following noun phrase and a prepositional phrase, the preposition usually attaches to the lower noun phrase. Similarly, prepositions usually attach to nearby noun phrases when they should attach to verbs farther to the left.

We see a more serious casualty of the dependency length bounds with conjunctions. In ground truth test data, 23 conjunctions are attached to $ and 141 to non-$ to using the COORD relation, whereas 100 conjunctions are attached to $ and 67 to non-$ using the AUXY relation. Our system overgeneralizes and attaches 84% of COORD and 71% of AUXY relations to $. Overall, conjunctions account for 15% of our errors. The AUXY relation is defined as "auxiliary (in compound expressions of various kinds)"; in the data, it seems to be often used for *waw*-consecutive or paratactic chaining of narrative clauses. If the conjunction *wa* ('and') begins a sentence, then that conjunction is tagged in ground truth as attaching to $; if the conjunction appears in the middle of the sentence, it may or may not be attached to $.

Noun attachments exhibit a more subtle problem. The direction of system attachments is biased more strongly to the left than is the case for the true data. In canonical order, Arabic nouns do generally attach on the right: subjects and objects follow the verb; in construct, the governed noun follows its governor. When the data deviate from this canonical order—when, e.g, a subject precedes its verb—the system prefers to find some other attachment point to the left. Similarly, a noun to the left of a conjunction often erroneously attaches to its left. Such ATR relations account for 35% of noun-attachment errors.

8 Conclusion

The tradeoff between speed and accuracy is familiar to any parsing researcher. Rather than starting with an accurate system and then applying corpus-specific speedups, we start by imposing carefully-chosen constraints (projectivity and length bounds) for speed, leaving accuracy to the parsing and reranking models. As it stands, our system performs poorly, largely because the estimation is not state-of-the-art, but also in part due to dependency length bounds, which are rather coarse at present. Better results are achievable by picking different bounds for different head tags (Eisner and N. Smith, 2005). Accuracy should not be difficult to improve using better learning methods, especially given our models' linear-time inference and decoding.

References

H. Alshawi. 1996. Head automata and bilingual tiling: Translation with minimal representations. In *Proc. of ACL*.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. of ICML*.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proc. of ACL*.

J. Eisner and N. A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *Proc. of IWPT*.

J. Eisner, E. Goldlust, and N. A. Smith. 2004. Dyna: A declarative language for implementing dynamic programs. In *Proc. of ACL* (companion volume).

J. Eisner. 1997. Bilexical grammars and a cubic-time probabilistic parser. In *Proc. of IWPT*.

L. Huang and D. Chiang. 2005. Better k-best parsing. In *Proc. of IWPT*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

K. Rose. 1998. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proc. of the IEEE*, 86(11):2210–2239.

D. A. Smith and J. Eisner. 2006. Minimum risk annealing for training log-linear models. To appear in *Proc. of COLING-ACL*.

Investigating Multilingual Dependency Parsing

Richard Johansson
Department of Computer Science
LTH, Lund University
221 00 Lund, Sweden
Richard.Johansson@cs.lth.se

Pierre Nugues
Department of Computer Science
LTH, Lund University
221 00 Lund, Sweden
Pierre.Nugues@cs.lth.se

Abstract

In this paper, we describe a system for the CoNLL-X shared task of multilingual dependency parsing. It uses a baseline Nivre's parser (Nivre, 2003) that first identifies the parse actions and then labels the dependency arcs. These two steps are implemented as SVM classifiers using LIBSVM. Features take into account the static context as well as relations dynamically built during parsing.

We experimented two main additions to our implementation of Nivre's parser: N-best search and bidirectional parsing. We trained the parser in both left-right and right-left directions and we combined the results. To construct a single-head, rooted, and cycle-free tree, we applied the Chu-Liu/Edmonds optimization algorithm. We ran the same algorithm with the same parameters on all the languages.

1 Nivre's Parser

Nivre (2003) proposed a dependency parser that creates a projective and acyclic graph. The parser is an extension to the shift–reduce algorithm. As with the regular shift–reduce, it uses a stack S and a list of input words W. However, instead of finding constituents, it builds a set of arcs G representing the graph of dependencies.

Nivre's parser uses two operations in addition to shift and reduce: left-arc and right-arc. Given a sequence of words, possibly annotated with their part of speech, parsing simply consists in applying a sequence of operations: left-arc (`la`), right-arc (`ra`), reduce (`re`), and shift (`sh`) to the input sequence.

2 Parsing an Annotated Corpus

The algorithm to parse an annotated corpus is straightforward from Nivre's parser and enables us to obtain, for any projective sentence, a sequence of actions taken in the set $\{\texttt{la}, \texttt{ra}, \texttt{re}, \texttt{sh}\}$ that parses it. At a given step of the parsing process, let TOP be the top of the stack and $FIRST$, the first token of the input list, and arc, the relation holding between a head and a dependent.

1. **if** $arc(TOP, FIRST) \in G$, **then** `ra`;
2. **else if** $arc(FIRST, TOP) \in G$, **then** `la`;
3. **else if** $\exists k \in Stack, arc(FIRST, k) \in G$ or $arc(k, FIRST) \in G$, **then** `re`;
4. **else** `sh`.

Using the first sentence of the Swedish corpus as input (Table 1), this algorithm produces the sequence of 24 actions: `sh`, `sh`, `la`, `ra`, `re`, `la`, `sh`, `sh`, `sh`, `la`, `la`, `ra`, `ra`, `sh`, `la`, `re`, `ra`, `ra`, `ra`, `re`, `re`, `re`, `re`, and `ra` (Table 2).

3 Adapting Nivre's Algorithm to Machine–Learning

3.1 Overview

We used support vector machines to predict the parse action sequence and a two step procedure to

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 206–210, New York City, June 2006.

Table 1: Dependency graph of the sentence *Äktenskapet och familjen är en gammal institution, som funnits sedan 1800-talet* 'Marriage and family are an old institution that has been around from the 19th century'.

ID	Form	POS	Head	Rel.
1	Äktenskapet	NN	4	SS
2	och	++	3	++
3	familjen	NN	1	CC
4	är	AV	0	ROOT
5	en	EN	7	DT
6	gammal	AJ	7	AT
7	institution	NN	4	SP
8	,	IK	7	IK
9	som	PO	10	SS
10	funnits	VV	7	ET
11	sedan	PR	10	TA
12	1800-talet	NN	11	PA
13	.	IP	4	IP

produce the graph. We first ran the classifier to select unlabeled actions, `la`, `ra`, `sh`, `re`. We then ran a second classifier to assign a function to `ra` and `la` parse actions.

We used the LIBSVM implementation of the SVM learning algorithm (Chang and Lin, 2001). We used the Gaussian kernel throughout. Optimal values for the parameters (C and γ) were found using a grid search. The first predicted action is not always possible, given the parser's constraints. We trained the model using probability estimates to select the next possible action.

3.2 Feature Set

We used the following set of features for the classifiers:

- Word and POS of *TOP* and *FIRST*
- Word and POS of the second node on the stack
- Word and POS of the second node in the input list
- POS of the third and fourth nodes in the input list
- The dependency type of *TOP* to its head, if any
- The word, POS, and dependency type of the leftmost child of *TOP* to *TOP*, if any
- The word, POS, and dependency type of the rightmost child of *TOP* to *TOP*, if any
- The word, POS, and dependency type of the leftmost child of *FIRST* to *FIRST*, if any

For the POS, we used the Coarse POS, the Fine POS, and all the features (encoded as boolean flags). We did not use the lemma.

Table 2: Actions to parse the sentence *Äktenskapet och familjen är en gammal institution, som funnits sedan 1800-talet*.

Ac.	Top word	First word	Rel.
sh	*nil*	Äktenskapet	
sh	Äktenskapet	och	
la	och	familjen	++
ra	Äktenskapet	familjen	CC
re	familjen	är	
la	Äktenskapet	är	SS
sh	nil	är	
sh	är	en	
sh	en	gammal	
la	gammal	institution	AT
la	en	institution	DT
ra	är	institution	SP
ra	institution	,	IK
sh	,	som	
la	som	funnits	SS
re	,	funnits	
ra	institution	funnits	ET
ra	funnits	sedan	TA
ra	sedan	1800-talet	PA
re	1800-talet	.	
re	sedan	.	
re	funnits	.	
re	institution	.	
ra	är	.	IP

4 Extensions to Nivre's Algorithm

4.1 N-best Search

We extended Nivre's original algorithm with a beam search strategy. For each action, `la`, `ra`, `sh` and `re`,

we computed a probability score using LIBSVM. These scores can then be used to carry out an N-best search through the set of possible sequences of actions.

We measured the improvement over a best-first strategy incrementing values of N. We observed the largest difference between N = 1 and N = 2, then leveling off and we used the latter value.

4.2 Bidirectionality and Voting

Tesnière (1966) classified languages as centrifuge (head to the left) and centripetal (head to the right) in a table (page 33 of his book) that nearly exactly fits corpus evidence from the CONLL data. Nivre's parser is inherently left-right. This may not fit all the languages. Some dependencies may be easier to capture when proceeding from the reverse direction. Jin et al. (2005) is an example of it for Chinese, where the authors describe an adaptation of Nivre's parser to bidirectionality.

We trained the model and ran the algorithm in both directions (left to right and right to left). We used a voting strategy based on probability scores. Each link was assigned a probability score (simply by using the probability of the `la` or `ra` actions for each link). We then summed the probability scores of the links from all four trees. To construct a single-head, rooted, and cycle-free tree, we finally applied the Chu-Liu/Edmonds optimization algorithm (Chu and Liu, 1965; Edmonds, 1967).

5 Analysis

5.1 Experimental Settings

We trained the models on "projectivized" graphs following Nivre and Nilsson (2005) method. We used the complete annotated data for nine langagues. Due to time limitations, we could not complete the training for three languages, Chinese, Czech, and German.

5.2 Overview of the Results

We parsed the 12 languages using exactly the same algorithms and parameters. We obtained an average score of 74.93 for the labeled arcs and of 80.39 for the unlabeled ones (resp. 74.98 and 80.80 for the languages where we could train the model using the complete annotated data sets). Table 3 shows the results per language. As a possible explanation of the differences between languages, the three lowest figures correspond to the three smallest corpora. It is reasonable to assume that if corpora would have been of equal sizes, results would have been more similar. Czech is an exception to this rule that applies to all the participants. We have no explanation for this. This language, or its annotation, seems to be more complex than the others.

The percentage of nonprojective arcs also seems to play a role. Due to time limitations, we trained the Dutch and German models with approximately the same quantity of data. While both languages are closely related, the Dutch corpus shows twice as much nonprojective arcs. The score for Dutch is significantly lower than for German.

Our results across the languages are consistent with the other participants' mean scores, where we are above the average by a margin of 2 to 3% except for Japanese and even more for Chinese where we obtain results that are nearly 7% less than the average for labeled relations. Results are similar for unlabeled data. We retrained the data with the complete Chinese corpus and you obtained 74.41 for the labeled arcs, still far from the average. We have no explanation for this dip with Chinese.

5.3 Analysis of Swedish and Portuguese Results

5.3.1 Swedish

We obtained a score of 78.13% for the labeled attachments in Swedish. The error breakdown shows significant differences between the parts of speech. While we reach 89% of correct head and dependents for the adjectives, we obtain 55% for the prepositions. The same applies to dependency types, 84% precision for subjects, and 46% for the OA type of prepositional attachment.

There is no significant score differences for the left and right dependencies, which could attributed to the bidirectional parsing (Table 4). Distance plays a dramatic role in the error score (Table 5). Prepositions are the main source of errors (Table 6).

5.3.2 Portuguese

We obtained a score 84.57% for the labeled attachments in Portuguese. As for Swedish, error distribution shows significant variations across the

Table 3: Summary of results. We retrained the Chinese* model after the deadline.

Languages	Unlabeled	Labeled
Completed training		
Arabic	75.53	64.29
Chinese*	79.13	74.41
Danish	86.59	81.54
Dutch	76.01	72.67
Japanese	87.11	85.63
Portuguese	88.4	84.57
Slovene	74.36	66.43
Spanish	81.43	78.16
Swedish	84.17	78.13
Turkish	73.59	63.39
$\overline{x}$	80.80	74.98
σ	5.99	8.63
Noncompleted training		
Chinese	77.04	72.49
Czech	77.4	71.46
German	83.09	80.43
$\overline{x}$ all languages	80.39	74.93
σ all languages	5.36	7.65

parts of speech, with a score of 94% for adjectives and only 67% for prepositions.

As for Swedish, there is no significant score differences for the left and right dependencies (Table 7). Distance also degrades results but the slope is not as steep as with Swedish (Table 8). Prepositions are also the main source of errors (Table 9).

5.4 Acknowledgments

This work was made possible because of the annotated corpora that were kindly provided to us: Arabic (Hajič et al., 2004), Bulgarian (Simov et al., 2005; Simov and Osenova, 2003), Chinese (Chen et al., 2003), Czech (Böhmová et al., 2003), Danish (Kromann, 2003), Dutch (van der Beek et al., 2002), German (Brants et al., 2002), Japanese (Kawata and Bartels, 2000), Portuguese (Afonso et al., 2002), Slovene (Džeroski et al., 2006), Spanish (Civit Torruella and Martí Antonín, 2002), Swedish (Nilsson et al., 2005), and Turkish (Oflazer et al., 2003; Atalay et al., 2003).

Table 4: Precision and recall of binned HEAD direction. Swedish.

Dir.	Gold	Cor.	Syst.	R	P
to_root	389	330	400	84.83	82.50
left	2745	2608	2759	95.01	94.53
right	1887	1739	1862	92.16	93.39

Table 5: Precision and recall of binned HEAD distance. Swedish.

Dist.	Gold	Cor.	Syst.	R	P
to_root	389	330	400	84.83	82.50
1	2512	2262	2363	90.05	95.73
2	1107	989	1122	89.34	88.15
3-6	803	652	867	81.20	75.20
7-...	210	141	269	67.14	52.42

Table 6: Focus words where most of the errors occur. Swedish.

Word	POS	Any	Head	Dep	Both
till	PR	48	20	45	17
i	PR	42	25	34	17
på	PR	39	22	32	15
med	PR	28	11	25	8
för	PR	27	22	25	20

Table 7: Precision and recall of binned HEAD direction. Portuguese.

Dir.	Gold	Cor.	Syst.	R	P
to_root	288	269	298	93.40	90.27
left	3006	2959	3020	98.44	97.98
right	1715	1649	1691	96.15	97.52

Table 8: Precision and recall of binned HEAD distance. Portuguese.

Dist.	Gold	Cor.	Syst.	R	P
to_root	288	269	298	93.40	90.27
1	2658	2545	2612	95.75	97.43
2	1117	1013	1080	90.69	93.80
3-6	623	492	647	78.97	76.04
7-...	323	260	372	80.50	69.89

Table 9: Focus words where most of the errors occur. Portuguese.

Word	POS	Any	Head	Dep	Both
em	prp	66	38	47	19
de	prp	51	37	35	21
a	prp	46	30	39	23
e	conj	28	28	0	0
para	prp	21	13	18	10

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*. Kluwer Academic Publishers, Dordrecht.

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. "Floresta sintá(c)tica": a treebank for Portuguese. In *Proc. of the Third Intern. Conf. on Language Resources and Evaluation (LREC)*, pages 1698–1703.

N. B. Atalay, K. Oflazer, and B. Say. 2003. The annotation process in the Turkish treebank. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proc. of the First Workshop on Treebanks and Linguistic Theories (TLT)*.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

M. Civit Torruella and M^{a} A. Martí Antonín. 2002. Design principles for a Spanish treebank. In *Proc. of the First Workshop on Treebanks and Linguistic Theories (TLT)*.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proc. of the Fifth Intern. Conf. on Language Resources and Evaluation (LREC)*.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

Meixun Jin, Mi-Young Kim, and Jong-Hyeok Lee. 2005. Two-phase shift-reduce deterministic dependency parser of Chinese. In *Proceedings of the Second International Joint Conference on Natural Language Processing*.

Y. Kawata and J. Bartels. 2000. Stylebook for the Japanese treebank in VERBMOBIL. Verbmobil-Report 240, Seminar für Sprachwissenschaft, Universität Tübingen.

M. T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In *Proc. of the Second Workshop on Treebanks and Linguistic Theories (TLT)*.

J. Nilsson, J. Hall, and J. Nivre. 2005. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. In *Proc. of the NODALIDA Special Session on Treebanks*.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, June.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160, Nancy, 23-25 April.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15.

K. Simov and P. Osenova. 2003. Practical annotation scheme for an HPSG treebank of Bulgarian. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*, pages 17–24.

K. Simov, P. Osenova, A. Simov, and M. Kouylekov. 2005. Design and implementation of the Bulgarian HPSG-based treebank. In *Journal of Research on Language and Computation – Special Issue*, pages 495–522. Kluwer Academic Publishers.

Lucien Tesnière. 1966. *Éléments de syntaxe structurale*. Klincksieck, Paris, 2e edition.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

Dependency Parsing Based on Dynamic Local Optimization

Ting Liu Jinshan Ma Huijia Zhu Sheng Li
Information Retrieval Lab
Harbin Institute of Technology
Harbin, 150001, China
{tliu,mjs,hjzhu,ls}@ir.hit.edu.cn

Abstract

This paper presents a deterministic parsing algorithm for projective dependency grammar. In a bottom-up way the algorithm finds the local optimum dynamically. A constraint procedure is made to use more structure information. The algorithm parses sentences in linear time and labeling is integrated with the parsing. This parser achieves 63.29% labeled attachment score on the average in CoNLL-X Shared Task.

1 Introduction

Recently, dependency grammar has gained renewed attention in the parsing community. Good results have been achieved in some dependency parsers (Yamada and Matsumoto, 2003; Nivre et al., 2004). With the availability of many dependency treebanks (van der Beek et al., 2002; Hajič et al., 2004; Böhmová et al., 2003; Kromann, 2003; Džeroski et al., 2006) and more other treebanks which can be converted to dependency annotation (Brants et al., 2002; Nilsson et al., 2005; Chen et al., 2003; Kawata and Bartels, 2000), multi-lingual dependency parsing is proposed in CoNLL shared task (Buchholz et al., 2006).

Many previous works focus on unlabeled parsing, in which exhaustive methods are often used (Eisner, 1996). Their global searching performs well in the unlabeled dependency parsing. But with the increase of parameters, efficiency has to be considered in labeled dependency parsing. Thus deterministic parsing was proposed as a robust and efficient method in recent years. Such method breaks the construction of dependency tree into a series of actions. A classifier is often used to choose the most probable action to assemble the dependency tree. (Yamada and Matsumoto, 2003) defined three actions and used a SVM classifier to choose one of them in a bottom-up way. The algorithm in (Nivre et al., 2004) is a blend of bottom-up and top-down processing. Its classifier is trained by memory-based learning.

Deterministic parsing derives an analysis without redundancy or backtracking, and linear time can be achieved. But when searching the local optimum in the order of left-to-right, some wrong reduce may prevent next analysis with more possibility. (Jin et al., 2005) used a two-phase shift-reduce to decrease such errors, and improved the accuracy of long distance dependencies.

In this paper a deterministic parsing based on dynamic local optimization is proposed. According to the probabilities of dependency arcs, the algorithm dynamically finds the one with the highest probabilities instead of dealing with the sentence in order. A procedure of constraint which can integrate more structure information is made to check the rationality of the reduce. Finally our results and error analysis are presented.

2 Dependency Probabilities

An example of Chinese dependency tree is showed in Figure1. The tree can be represented as a directed graph with nodes representing word tokens and arcs

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 211–215, New York City, June 2006.

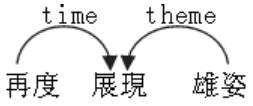


Figure 1: A Chinese dependency tree

representing dependency relations. The assumption that the arcs are independent on each other often is made so that parsing can be handled easily. On the other side the independence assumption will result in the loss of information because dependencies are interrelated on each other actually. Therefore, two kinds of probabilities are used in our parser. One is arc probabilities which are the possibility that two nodes form an arc, and the other is structure probabilities which are used to describe some specific syntactic structures.

2.1 Arc Probabilities

A dependency arc A_i can be expressed as a 4-tuple $A_i = <Node_i, Node_j, D, R>$. $Node_i$ and $Node_j$ are nodes that constitute the directed arc. D is the direction of the arc, which can be *left* or *right*. R is relation type labeled on the arc. Under the independence assumption that an arc depends on its two nodes we can calculate arc probability given two nodes. In our paper the arc probabilities are calculated as follows:

$P_1 = P(R,D|CTag_i, CTag_j, Dist)$
$P_2 = P(R,D|FTag_i, FTag_j)$
$P_3 = P(R,D|CTag_i, Word_j)$
$P_4 = P(R,D|Word_i, CTag_j)$
$P_5 = P(R,D|Word_i, CTag_i, Word_j, CTag_j)$
$P_6 = P(R,D|CTag_{i-1}, CTag_i, CTag_j, CTag_{j+1})$

Where *CTag* is coarse-grained part of speech tag and *FTag* is fine-grained tag. As to *Word* we choose its lemma if it exists. *Dist* is the distance between $Node_i$ and $Node_j$. It is divided into four parts:

$Dist = 1 \quad if \quad j\text{-}i = 1$
$Dist = 2 \quad if \quad j\text{-}i = 2$
$Dist = 3 \quad if \quad 3 \leqslant j\text{-}i \leqslant 6$
$Dist = 4 \quad if \quad j\text{-}i > 6$

All the probabilities are obtained by maximum likelihood estimation from the training data. Then interpolation smoothing is made to get the final arc probabilities.

2.2 Structure Probabilities

Structure information plays the critical role in syntactic analysis. Nevertheless the flexibility of syntactic structures and data sparseness pose obstacles to us. Especially some structures are related to specific language and cannot be employed in multilingual parsing. We have to find those language-independent features.

In valency theory "valence" represents the number of arguments that a verb is able to govern. In this paper we extend the range of verbs and arguments to all the words. We call the new "valence" *Governing Degree* (GD), which means the ability of one node governing other nodes. In Figure1, the GD of node "展現" is 2 and the GDs of two other nodes are 0. The governing degree of nodes in dependency tree often shows directionality. For example, Chinese token "的" always governs one left node. Furthermore, we subdivide the GD into *Left Governing Degree* (LGD) and *Right Governing Degree* (RGD), which are the ability of words governing their left children or right children. In Figure 1 the LGD and RGD of verb "展現" are both 1.

In the paper we use the probabilities of GD over the fine-grained tags. The probabilities of *P(LDG|FTag)* and *P(RGD|FTag)* are calculated from training data. Then we only reserve the *FTags* with large probability because their GDs are stable and helpful to syntactic analysis. Other *FTags* with small probabilities are unstable in GDs and cannot provide efficient information for syntactic analysis. If their probabilities are less than 0.65 they will be ignored in our dependency parsing.

3 Dynamic local optimization

Many previous methods are based on history-based models. Despite many obvious advantages, these methods can be awkward to encode some constrains within their framework (Collins, 2000). Classifiers are good at encoding more features in the deterministic parsing (Yamada and Matsumoto, 2003; Nivre et al., 2004). However, such algorithm often make more probable dependencies be prevented by preceding errors. An example is showed in Figure 2. Arc *a* is a frequent dependency and *b* is an arc with more probability. Arc *b* will be prevented by *a* if the reduce is carried out in order.

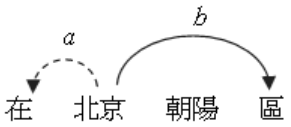


Figure 2: A common error in deterministic parsing

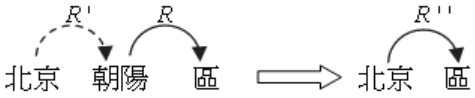


Figure 3: Adjustment

3.1 Our algorithm

Our deterministic parsing is based on dynamic local optimization. The algorithm calculates the arc probabilities of two continuous nodes, and then reduces the most probable arc. The construction of dependency tree includes four actions: **Check**, **Reduce**, **Delete**, and **Insert**. Before a node is reduced, the Check procedure is made to validate its correctness. Only if the arc passes the Check procedure it can be reduced. Otherwise the Reduce will be delayed. Delete and Insert are then carried out to adjust the changed arcs. The complete algorithm is depicted as follows:

```
Input Sentence: S = (w1, w2,    , wn)
Initialize:
for i = 1 to n
    Ri = GetArcProb(wi,wi+1);
    Push(Ri) onto Stack;
Sort(Stack);
Start:
i = 0;
While  Stack.empty = false
        R = Stack.top+i;
        if Check(R) = true
           Reduce(R);
           Delete(R');
           Insert(R");
           i = 0;
        else
           i++;
```

The algorithm has following advantages:

- Projectivity can be guaranteed. The node is only reduced with its neighboring node. If a node is reduced as a leaf it will be removed from the sentence and doesn't take part in next Reduce. So no cross arc will occur.
- After n-1 pass a projective dependency tree is complete. Algorithm is finished in linear time.
- The algorithm always reduces the node with the highest probability if it passes the Check. No any limitation on order thus the spread of errors can be mitigated effectively.
- Check is an open process. Various constrains can be encoded in this process. Structural constrains, partial parsed information or language-dependent knowledge can be added.

Adjustment is illustrated in Figure 3, where "朝陽" is reduced and arc *R'* is deleted. Then the algorithm computes the arc probability of *R"* and inserts it to the Stack.

3.2 Checking

The information in parsing falls into two kinds: static and dynamic. The arc probabilities in 2.1 describe the static information which is not changed in parsing. They are obtained from the training data in advance. The structure probabilities in 2.2 describe the dynamic information which varies in the process of parsing. The use of dynamic information often depends on what current dependency tree is.

Besides the governing degree, Check procedure also uses another dynamic information–*Sequential Dependency*. Whether current arc can be reduced is relating to previous arc. In Figure 3 the reduce of the arc *R* depends on the arc *R'*. If *R'* has been delayed or its probability is little less than that of *R*, arc *R* will be delayed.

If the arc doesn't pass the Check it will be delayed. The delayed time ranges from 1 to *Length* which is the length of sentence. If the arc is delayed *Length* times it will be blocked. The Reduce will be delayed in the following cases:

- $\widehat{GD}(Node_i) > 0$ and its probability is *P*. If GD($Node_i$) = 0 and $Node_i$ is made as child in the Reduce, the $Node_i$ will be delayed *Length*P* times.
- $\widehat{GD}(Node_i) \leqslant m$ ($m > 0$) and its probability is *P*. If GD($Node_i$) = *m* and $Node_i$ is made as parent in the Reduce, the $Node_i$ will be delayed *Length*P* times.

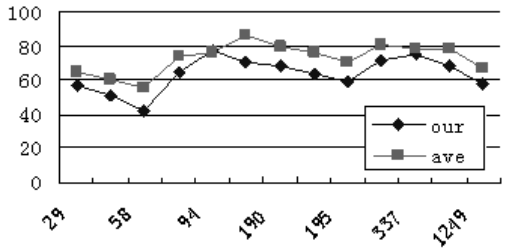


Figure 4: Token score with size of training data

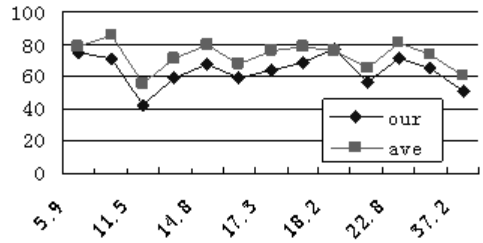


Figure 5: Token score with sentence length

- $P(R') > \lambda P(R)$, the current arc R will be delayed $Length*(P(R')/P(R))$ times. R' is the preceding arc and $\lambda = 0.60$.
- If arc R' is blocking, the arc R will be delayed.

$\widehat{GD}$ is empirical value and GD is current value.

4 Experiments and analysis

Our parsing results and average results are listed in the Table 1. It can be seen that the attachment scores vary greatly with different languages. A general analysis and a specific analysis are made respectively in this section.

4.1 General analysis

We try to find the properties that make the difference to parsing results in multi-lingual parsing. The properties of all the training data can be found in (Buchholz et al., 2006). Intuitively the size of training data and average length of per sentence would be influential on dependency parsing. The relation of these properties and scores are showed in the Figure 4 and 5.

From the charts we cannot assuredly find the properties that are proportional to score. Whether Czech language with the largest size of training data or Chinese with the shortest sentence length, don't achieve the best results. It seems that no any factor is determining to parsing results but all the properties exert influence on the dependency parsing together.

Another factor that maybe explain the difference of scores in multi-lingual parsing is the characteristics of language. For example, the number of tokens with HEAD=0 in a sentence is not one for some languages. Table 1 shows the range of governing degree of head. This statistics is somewhat different with that from organizers because we don't distinguish the scoring tokens and non-scoring tokens.

Another characteristic is the directionality of dependency relations. As Table 1 showed, many relations in treebanks are bi-directional, which increases the number of the relation actually. Furthermore, the flexibility of some grammatical structures poses difficulties to language model. For instance, subject can appear in both sides of the predicates in some treebanks which tends to cause the confusion with the object (Kromann, 2003; Afonso et al., 2002; Civit Torruella and Martí Antonín, 2002; Oflazer et al., 2003; Atalay et al., 2003).

As to our parsing results, which are lower than all the average results except for Danish. That can be explained from the following aspects:

(1) Our parser uses a projective parsing algorithm and cannot deal with the non-projective tokens, which exist in all the languages except for Chinese.

(2) The information provided by training data is not fully employed. Only POS and lemma are used. The morphological and syntactic features may be helpful to parsing.

(3) We haven't explored syntactic structures in depth for multi-lingual parsing and more structural features need to be used in the Check procedure.

4.2 Specific analysis

Specifically we make error analysis to Chinese and Turkish. In Chinese result we found many errors occurred near the auxiliary word "的"(DE). We call the noun phrases with "的" *DE Structure*. The word "的" appears 355 times in the all 4970 dependencies of the test data. In Table 2 the second row shows the frequencies of "DE" as the parent of dependencies. The third row shows the frequencies while it is as child. Its error rate is 33.1% and 43.4% in our results respectively. Furthermore, each head error will result in more than one errors, so the errors from *DE Structures* are nearly 9% in our results.

	Ar	Ch	Cz	Da	Du	Ge	Ja	Po	Sl	Sp	Sw	Tu
our	50.74	75.29	58.52	77.70	59.36	68.11	70.84	71.13	57.21	65.08	63.83	41.72
ave	59.94	78.32	67.17	76.16	70.73	78.58	85.86	80.63	65.16	73.52	76.44	55.95
NH	17	1	28	4	9	1	14	1	11	1	1	5
BD	27/24	78/55	82/72	54/24	26/17	46/40	7/2	55/40	26/23	21/19	64/54	26/23

Table 1: The second and third rows are our scores and average scores. The fourth row lists the maximal number of tokens with HEAD=0 in a sentence. The last row lists the number of relations/the number of bi-directional relations of them (Our statistics are slightly different from that of organizers).

	gold	system	error	headerr
parent	320	354	106	106
child	355	355	154	74

Table 2: Chinese DE Structure Errors

	total	obj	sub	mod	D.A	L.A
Noun-V	1300	494	319	156	102	78
Pron-V	215	91	60	9	37	3

Table 3: The distribution of some common relations

The high error rate is due to the flexibility of *DE Structure*. The children of DE can be nouns and verbs, thus the ambiguities will occur. For example, the sequence "V N1 DE N2" is a common ambiguious structure in Chinese. It needs to be solved with semantic knowledge to some extent. The errors of DE being child are mostly from noun compounds. For example, the string "週詳 的 作品 維護" results in the error: "DE" as the child of "作品". It will be better that noun compounds are processed specially.

Our results and average results achieve the lowest score on Turkish. We try to find some reasons through the following analysis. Turkish is a typical head-final language and 81.1% of dependencies are right-headed. The monotone of directionality increases the difficulties of identification. Another difficulty is the diversity of the same pair. Taking noun and pronoun as example, which only achieve the accuracy of 25% and 28% in our results, there are 14 relations in the noun-verb pairs and 11 relations in the pronoun-verb pairs. Table 3 illustrates the distribution of some common relations in the test data.

The similarity of these dependencies makes our parser only recognize 23.3% noun-verb structures and 21.8% pronoun-verb structures. The syntactic or semantic knowledge maybe helpful to distinguish these similar structures.

5 Conclusion

This paper has applied a deterministic algorithm based on dynamic local optimization to multi-lingual dependency parsing. Through the error analysis for some languages, we think that the difference between languages is a main obstacle posed on multi-lingual dependency parsing. Adopting different learners according to the type of languages may be helpful to multi-lingual dependency parsing.

Acknowledgement This work was supported by the National Natural Science Foundation of China under Grant No. 60435020，60575042 and 60503072.

References

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. of ICML*.

M.X. Jin, M.Y. Kim, and J.H. Lee. 2005. Two-phase shift-reduce deterministic dependency parser of chinese. In *Proc. of IJCNLP: Companion Volume including Posters/Demos and tutorial abstracts*.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proc. of the Eighth Conf. on Computational Natural Language Learning (CoNLL)*, pages 49–56.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of the 16th Intern. Conf. on Computational Linguistics (COLING)*, pages 340–345.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of the 8th Intern. Workshop on Parsing Technologies (IWPT)*.

Multilingual Dependency Analysis with a Two-Stage Discriminative Parser

Ryan McDonald **Kevin Lerman** **Fernando Pereira**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA
{ryantm,klerman,pereira}@cis.upenn.edu

Abstract

We present a two-stage multilingual dependency parser and evaluate it on 13 diverse languages. The first stage is based on the unlabeled dependency parsing models described by McDonald and Pereira (2006) augmented with morphological features for a subset of the languages. The second stage takes the output from the first and labels all the edges in the dependency graph with appropriate syntactic categories using a globally trained sequence classifier over components of the graph. We report results on the CoNLL-X shared task (Buchholz et al., 2006) data sets and present an error analysis.

1 Introduction

Often in language processing we require a deep syntactic representation of a sentence in order to assist further processing. With the availability of resources such as the Penn WSJ Treebank, much of the focus in the parsing community had been on producing syntactic representations based on phrase-structure. However, recently their has been a revived interest in parsing models that produce dependency graph representations of sentences, which model words and their arguments through directed edges (Hudson, 1984; Mel′čuk, 1988). This interest has generally come about due to the computationally efficient and flexible nature of dependency graphs and their ability to easily model non-projectivity in freer-word order languages. Nivre (2005) gives an introduction to dependency representations of sentences and recent developments in dependency parsing strategies.

Dependency graphs also encode much of the deep syntactic information needed for further processing. This has been shown through their successful use in many standard natural language processing tasks, including machine translation (Ding and Palmer, 2005), sentence compression (McDonald, 2006), and textual inference (Haghighi et al., 2005).

In this paper we describe a two-stage discriminative parsing approach consisting of an unlabeled parser and a subsequent edge labeler. We evaluate this parser on a diverse set of 13 languages using data provided by the CoNLL-X shared-task organizers (Buchholz et al., 2006; Hajič et al., 2004; Simov et al., 2005; Simov and Osenova, 2003; Chen et al., 2003; Böhmová et al., 2003; Kromann, 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit Torruella and Martí Antonín, 2002; Nilsson et al., 2005; Oflazer et al., 2003; Atalay et al., 2003). The results are promising and show the language independence of our system under the assumption of a labeled dependency corpus in the target language.

For the remainder of this paper, we denote by $\boldsymbol{x} = x_1, \ldots x_n$ a sentence with n words and by $\boldsymbol{y}$ a corresponding dependency graph. A dependency graph is represented by a set of ordered pairs $(i, j) \in \boldsymbol{y}$ in which x_j is a dependent and x_i is the corresponding head. Each edge can be assigned a label $l_{(i,j)}$ from a finite set L of predefined labels. We

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 216–220, New York City, June 2006.

assume that all dependency graphs are trees but may be non-projective, both of which are true in the data sets we use.

2 Stage 1: Unlabeled Parsing

The first stage of our system creates an unlabeled parse $\boldsymbol{y}$ for an input sentence $\boldsymbol{x}$. This system is primarily based on the parsing models described by McDonald and Pereira (2006). That work extends the maximum spanning tree dependency parsing framework (McDonald et al., 2005a; McDonald et al., 2005b) to incorporate features over multiple edges in the dependency graph. An exact projective and an approximate non-projective parsing algorithm are presented, since it is shown that non-projective dependency parsing becomes NP-hard when features are extended beyond a single edge.

That system uses MIRA, an online large-margin learning algorithm, to compute model parameters. Its power lies in the ability to define a rich set of features over parsing decisions, as well as surface level features relative to these decisions. For instance, the system of McDonald et al. (2005a) incorporates features over the part of speech of words occurring between and around a possible head-dependent relation. These features are highly important to overall accuracy since they eliminate unlikely scenarios such as a preposition modifying a noun not directly to its left, or a noun modifying a verb with another verb occurring between them.

We augmented this model to incorporate morphological features derived from each token. Consider a proposed dependency of a dependent x_j on the head x_i, each with morphological features M_j and M_i respectively. We then add to the representation of the edge: M_i as head features, M_j as dependent features, and also each conjunction of a feature from both sets. These features play the obvious role of explicitly modeling consistencies and commonalities between a head and its dependents in terms of attributes like gender, case, or number. Not all data sets in our experiments include morphological features, so we use them only when available.

3 Stage 2: Label Classification

The second stage takes the output parse $\boldsymbol{y}$ for sentence $\boldsymbol{x}$ and classifies each edge $(i,j) \in \boldsymbol{y}$ with a particular label $l_{(i,j)}$. Ideally one would like to make all parsing and labeling decisions jointly so that the shared knowledge of both decisions will help resolve any ambiguities. However, the parser is fundamentally limited by the scope of local factorizations that make inference tractable. In our case this means we are forced only to consider features over single edges or pairs of edges. However, in a two stage system we can incorporate features over the entire output of the unlabeled parser since that structure is fixed as input. The simplest labeler would be to take as input an edge $(i,j) \in \boldsymbol{y}$ for sentence $\boldsymbol{x}$ and find the label with highest score,

$$l_{(i,j)} = \arg\max_{l} \; s(l, (i,j), \boldsymbol{y}, \boldsymbol{x})$$

Doing this for each edge in the tree would produce the final output. Such a model could easily be trained using the provided training data for each language. However, it might be advantageous to know the labels of other nearby edges. For instance, if we consider a head x_i with dependents $x_{j_1}, \ldots, x_{j_M}$, it is often the case that many of these dependencies will have correlated labels. To model this we treat the labeling of the edges $(i,j_1), \ldots, (i,j_M)$ as a sequence labeling problem,

$$(l_{(i,j_1)}, \ldots, l_{(i,j_M)}) = \bar{l} = \arg\max_{\bar{l}} \; s(\bar{l}, i, \boldsymbol{y}, \boldsymbol{x})$$

We use a first-order Markov factorization of the score

$$\bar{l} = \arg\max_{\bar{l}} \; \sum_{m=2}^{M} s(l_{(i,j_m)}, l_{(i,j_{m-1})}, i, \boldsymbol{y}, \boldsymbol{x})$$

in which each factor is the score of labeling the adjacent edges (i,j_m) and (i,j_{m-1}) in the tree $\boldsymbol{y}$. We attempted higher-order Markov factorizations but they did not improve performance uniformly across languages and training became significantly slower.

For score functions, we use simple dot products between high dimensional feature representations and a weight vector

$$s(l_{(i,j_m)}, l_{(i,j_{m-1})}, i, \boldsymbol{y}, \boldsymbol{x}) = \\ \mathbf{w} \cdot \mathbf{f}(l_{(i,j_m)}, l_{(i,j_{m-1})}, i, \boldsymbol{y}, \boldsymbol{x})$$

Assuming we have an appropriate feature representation, we can find the highest scoring label sequence with Viterbi's algorithm. We use the MIRA

online learner to set the weights (Crammer and Singer, 2003; McDonald et al., 2005a) since we found it trained quickly and provide good performance. Furthermore, it made the system homogeneous in terms of learning algorithms since that is what is used to train our unlabeled parser (McDonald and Pereira, 2006). Of course, we have to define a set of suitable features. We used the following:

- **Edge Features:** Word/pre-suffix/part-of-speech (POS)/morphological feature identity of the head and the dependent (affix lengths 2 and 3). Does the head and its dependent share a prefix/suffix? Attachment direction. What morphological features do head and dependent have the same value for? Is the dependent the first/last word in the sentence?

- **Sibling Features:** Word/POS/pre-suffix/morphological feature identity of the dependent's nearest left/right siblings in the tree (siblings are words with same parent in the tree). Do any of the dependent's siblings share its POS?

- **Context Features:** POS tag of each intervening word between head and dependent. Do any of the words between the head and the dependent have a parent other than the head? Are any of the words between the head and the dependent not a descendant of the head (i.e. non-projective edge)?

- **Non-local:** How many children does the dependent have? What morphological features do the grandparent and the dependent have identical values? Is this the left/rightmost dependent for the head? Is this the first dependent to the left/right of the head?

Various conjunctions of these were included based on performance on held-out data. Note that many of these features are beyond the scope of the edge based factorizations of the unlabeled parser. Thus a joint model of parsing and labeling could not easily include them without some form of re-ranking or approximate parameter estimation.

4 Results

We trained models for all 13 languages provided by the CoNLL organizers (Buchholz et al., 2006). Based on performance from a held-out section of the training data, we used non-projective parsing algorithms for Czech, Danish, Dutch, German, Japanese, Portuguese and Slovene, and projective parsing algorithms for Arabic, Bulgarian, Chinese, Spanish, Swedish and Turkish. Furthermore, for Arabic and Spanish, we used lemmas instead of inflected word forms, again based on performance on held-out data[1].

DATA SET	UA	LA
ARABIC	79.3	66.9
BULGARIAN	92.0	87.6
CHINESE	91.1	85.9
CZECH	87.3	80.2
DANISH	90.6	84.8
DUTCH	83.6	79.2
GERMAN	90.4	87.3
JAPANESE	92.8	90.7
PORTUGUESE	91.4	86.8
SLOVENE	83.2	73.4
SPANISH	86.1	82.3
SWEDISH	88.9	82.5
TURKISH	74.7	63.2
AVERAGE	87.0	80.8

Table 1: Dependency accuracy on 13 languages. Unlabeled (UA) and Labeled Accuracy (LA).

Results on the test set are given in Table 1. Performance is measured through unlabeled accuracy, which is the percentage of words that modify the correct head in the dependency graph, and labeled accuracy, which is the percentage of words that modify the correct head *and* label the dependency edge correctly in the graph. These results show that the discriminative spanning tree parsing framework (McDonald et al., 2005b; McDonald and Pereira, 2006) is easily adapted across all these languages. Only Arabic, Turkish and Slovene have parsing accuracies significantly below 80%, and these languages have relatively small training sets and/or are highly inflected with little to no word order constraints. Furthermore, these results show that a two-stage system can achieve a relatively high performance. In fact, for every language our models perform significantly higher than the average performance for all the systems reported in Buchholz et al. (2006).

For the remainder of the paper we provide a general error analysis across a wide set of languages plus a detailed error analysis of Spanish and Arabic.

5 General Error Analysis

Our system has several components, including the ability to produce non-projective edges, sequential

[1]Using the non-projective parser for all languages does not effect performance significantly. Similarly, using the inflected word form instead of the lemma for all languages does not change performance significantly.

SYSTEM	UA	LA
N+S+M	86.3	79.7
P+S+M	85.6	79.2
N+S+B	85.5	78.6
N+A+M	86.3	79.4
P+A+B	84.8	77.7

Table 2: Error analysis of parser components averaged over Arabic, Bulgarian, Danish, Dutch, Japanese, Portuguese, Slovene, Spanish, Swedish and Turkish. N/P: Allow non-projective/Force projective, S/A: Sequential labeling/Atomic labeling, M/B: Include morphology features/No morphology features.

assignment of edge labels instead of individual assignment, and a rich feature set that incorporates morphological properties when available. The benefit of each of these is shown in Table 2. These results report the average labeled and unlabeled precision for the 10 languages with the smallest training sets. This allowed us to train new models quickly.

Table 2 shows that each component of our system does not change performance significantly (rows 2-4 versus row 1). However, if we only allow projective parses, do not use morphological features and label edges with a simple atomic classifier, the overall drop in performance becomes significant (row 5 versus row 1). Allowing non-projective parses helped with freer word order languages like Dutch ($78.8\%/74.7\%$ to $83.6\%/79.2\%$, unlabeled/labeled accuracy). Including rich morphology features naturally helped with highly inflected languages, in particular Spanish, Arabic, Turkish, Slovene and to a lesser extent Dutch and Portuguese. Derived morphological features improved accuracy in all these languages by 1-3% absolute.

Sequential classification of labels had very little effect on overall labeled accuracy (79.4% to 79.7%)[2]. The major contribution was in helping to distinguish subjects, objects and other dependents of main verbs, which is the most common labeling error. This is not surprising since these edge labels typically are the most correlated (i.e., if you already know which noun dependent is the subject, then it should be easy to find the object). For instance, sequential labeling improves the labeling of objects from $81.7\%/75.6\%$ to $84.2\%/81.3\%$ (labeled precision/recall) and the labeling of subjects from $86.8\%/88.2\%$ to $90.5\%/90.4\%$ for Swedish. Similar improvements are common across all languages, though not as dramatic. Even with this improvement, the labeling of verb dependents remains the highest source of error.

[2]This difference was much larger for experiments in which gold standard unlabeled dependencies are used.

6 Detailed Analysis

6.1 Spanish

Although overall unlabeled accuracy is 86%, most verbs and some conjunctions attach to their head words with much lower accuracy: 69% for main verbs, 75% for the verb *ser*, and 65% for coordinating conjunctions. These words form 17% of the test corpus. Other high-frequency word classes with relatively low attachment accuracy are prepositions (80%), adverbs (82%) and subordinating conjunctions (80%), for a total of another 23% of the test corpus. These weaknesses are not surprising, since these decisions encode the more global aspects of sentence structure: arrangement of clauses and adverbial dependents in multi-clause sentences, and prepositional phrase attachment. In a preliminary test of this hypothesis, we looked at all of the sentences from a development set in which a main verb is incorrectly attached. We confirmed that the main clause is often misidentified in multi-clause sentences, or that one of several conjoined clauses is incorrectly taken as the main clause. To test this further, we added features to count the number of commas and conjunctions between a dependent verb and its candidate head. Unlabeled accuracy for all verbs increases from 71% to 73% and for all conjunctions from 71% to 74%. Unfortunately, accuracy for other word types decreases somewhat, resulting in no significant net accuracy change. Nevertheless, this very preliminary experiment suggests that wider-range features may be useful in improving the recognition of overall sentence structure.

Another common verb attachment error is a switch between head and dependent verb in phrasal verb forms like *dejan intrigar* or *qiero decir*, possibly because the non-finite verb in these cases is often a main verb in training sentences. We need to look more carefully at verb features that may be useful here, in particular features that distinguish finite and

non-finite forms.

In doing this preliminary analysis, we noticed some inconsistencies in the reference dependency structures. For example, in the test sentence *Lo que decia Mae West de si misma podríamos decirlo también los hombres:..., decia*'s head is given as *decirlo*, although the main verbs of relative clauses are normally dependent on what the relative modifies, in this case the article *Lo*.

6.2 Arabic

A quick look at unlabeled attachment accuracies indicate that errors in Arabic parsing are the most common across all languages: prepositions (62%), conjunctions (69%) and to a lesser extent verbs (73%). Similarly, for labeled accuracy, the hardest edges to label are for dependents of verbs, i.e., subjects, objects and adverbials. Note the difference in error between the unlabeled parser and the edge labeler: the former makes mistakes on edges into prepositions, conjunctions and verbs, and the latter makes mistakes on edges into nouns (subject/objects). Each stage by itself is relatively accurate (unlabeled accuracy is 79% and labeling accuracy[3] is also 79%), but since there is very little overlap in the kinds of errors each makes, overall labeled accuracy drops to 67%. This drop is not nearly as significant for other languages.

Another source of potential error is that the average sentence length of Arabic is much higher than other languages (around 37 words/sentence). However, if we only look at performance for sentences of length less than 30, the labeled accuracy is still only 71%. The fact that Arabic has only 1500 training instances might also be problematic. For example if we train on 200, 400, 800 and the full training set, labeled accuracies are 54%, 60%, 62% and 67%. Clearly adding more data is improving performance. However, when compared to the performance of Slovene (1500 training instances) and Spanish (3300 instances), it appears that Arabic parsing is lagging.

7 Conclusions

We have presented results showing that the spanning tree dependency parsing framework of McDonald et al. (McDonald et al., 2005b; McDonald and Pereira, 2006) generalizes well to languages other than English. In the future we plan to extend these models in two ways. First, we plan on examining the performance difference between two-staged dependency parsing (as presented here) and joint parsing plus labeling. It is our hypothesis that for languages with fine-grained label sets, joint parsing and labeling will improve performance. Second, we plan on integrating any available morphological features in a more principled manner. The current system simply includes all morphological bi-gram features. It is our hope that a better morphological feature set will help with both unlabeled parsing and labeling for highly inflected languages.

[3] Labeling accuracy is the percentage of words that correctly label the dependency between the head that they modify, even if the right head was not identified.

References

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. SIGNLL.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *JMLR*.

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. ACL*.

A. Haghighi, A. Ng, and C. Manning. 2005. Robust textual inference via graph matching. In *Proc. HTL-EMNLP*.

R. Hudson. 1984. *Word Grammar*. Blackwell.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. EACL*.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. ACL*.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. HLT-EMNLP*.

R. McDonald. 2006. Discriminative sentence compression with soft syntactic constraints. In *Proc. EACL*.

I.A. Mel′čuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

J. Nivre. 2005. Dependency grammar and dependency parsing. Technical Report MSI report 05133, Växjö University: School of Mathematics and Systems Engineering.

Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines

Joakim Nivre
Johan Hall
Jens Nilsson
School of Mathematics and Systems Engineering
Växjö University
35195 Växjö, Sweden
{nivre,jha,jni}@msi.vxu.se

Gülşen Eryiğit
Department of Computer Engineering
Istanbul Technical University
34469 Istanbul, Turkey
gulsen@cs.itu.edu.tr

Svetoslav Marinov
School of Humanities and Informatics
University of Skövde
Box 408
54128 Skövde, Sweden
svetoslav.marinov@his.se

Abstract

We use SVM classifiers to predict the next action of a deterministic parser that builds labeled projective dependency graphs in an incremental fashion. Non-projective dependencies are captured indirectly by projectivizing the training data for the classifiers and applying an inverse transformation to the output of the parser. We present evaluation results and an error analysis focusing on Swedish and Turkish.

1 Introduction

The CoNLL-X shared task consists in parsing texts in multiple languages using a single dependency parser that has the capacity to learn from treebank data. Our methodology for performing this task is based on four essential components:

- A deterministic algorithm for building labeled projective dependency graphs (Nivre, 2006).
- History-based feature models for predicting the next parser action (Black et al., 1992).
- Support vector machines for mapping histories to parser actions (Kudo and Matsumoto, 2002).
- Graph transformations for recovering non-projective structures (Nivre and Nilsson, 2005).

All experiments have been performed using MaltParser (Nivre et al., 2006), version 0.4, which is made available together with the suite of programs used for pre- and post-processing.[1]

[1] www.msi.vxu.se/users/nivre/research/MaltParser.html

2 Parsing Methodology

2.1 Parsing Algorithm

The parsing algorithm used for all languages is the deterministic algorithm first proposed for unlabeled dependency parsing by Nivre (2003) and extended to labeled dependency parsing by Nivre et al. (2004). The algorithm builds a labeled dependency graph in one left-to-right pass over the input, using a stack to store partially processed tokens and adding arcs using four elementary actions (where *top* is the token on top of the stack and *next* is the next token):

- SHIFT: Push *next* onto the stack.
- REDUCE: Pop the stack.
- RIGHT-ARC(r): Add an arc labeled r from *top* to *next*; push *next* onto the stack.
- LEFT-ARC(r): Add an arc labeled r from *next* to *top*; pop the stack.

Although the parser only derives projective graphs, the fact that graphs are labeled allows non-projective dependencies to be captured using the pseudo-projective approach of Nivre and Nilsson (2005) .

Another limitation of the parsing algorithm is that it does not assign dependency labels to roots, i.e., to tokens having HEAD=0. To overcome this problem, we have implemented a variant of the algorithm that starts by pushing an artificial root token with ID=0 onto the stack. Tokens having HEAD=0 can now be attached to the artificial root in a RIGHT-ARC(r) action, which means that they can be assigned any label. Since this variant of the algorithm increases the overall nondeterminism, it has only been used for the data sets that include informative root labels (Arabic, Czech, Portuguese, Slovene).

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X), pages 221–225, New York City, June 2006.

	FO	L	C	P	FE	D
S: *top*	+	+	+	+	+	+
S: *top*−1				+		
I: *next*	+	+	+	+	+	
I: *next*+1	+			+		
I: *next*+2				+		
I: *next*+3				+		
G: head of *top*	+					
G: leftmost dep of *top*						+
G: rightmost dep of *top*						+
G: leftmost dep of *next*						+

Table 1: Base model; S: stack, I: input, G: graph; FO: FORM, L: LEMMA , C: CPOS, P: POS, FE: FEATS, D: DEPREL

2.2 History-Based Feature Models

History-based parsing models rely on features of the derivation history to predict the next parser action. The features used in our system are all symbolic and extracted from the following fields of the data representation: FORM, LEMMA, CPOSTAG, POSTAG, FEATS, and DEPREL. Features of the type DEPREL have a special status in that they are extracted during parsing from the partially built dependency graph and may therefore contain errors, whereas all the other features have gold standard values during both training and parsing.[2]

Based on previous research, we defined a base model to be used as a starting point for language-specific feature selection. The features of this model are shown in Table 1, where rows denote tokens in a parser configuration (defined relative to the stack, the remaining input, and the partially built dependency graph), and where columns correspond to data fields. The base model contains twenty features, but note that the fields LEMMA, CPOS and FEATS are not available for all languages.

2.3 Support Vector Machines

We use support vector machines[3] to predict the next parser action from a feature vector representing the history. More specifically, we use LIBSVM (Chang and Lin, 2001) with a quadratic kernel $K(x_i, x_j) = (\gamma x_i^T x_j + r)^2$ and the built-in one-versus-all strategy for multi-class classification. Symbolic features are converted to numerical features using the standard technique of binarization, and we split values of the FEATS field into its atomic components.[4]

For some languages, we divide the training data into smaller sets, based on some feature s (normally the CPOS or POS of the next input token), which may reduce training times without a significant loss in accuracy (Yamada and Matsumoto, 2003). To avoid too small training sets, we pool together categories that have a frequency below a certain threshold t.

2.4 Pseudo-Projective Parsing

Pseudo-projective parsing was proposed by Nivre and Nilsson (2005) as a way of dealing with non-projective structures in a projective data-driven parser. We projectivize training data by a minimal transformation, lifting non-projective arcs one step at a time, and extending the arc label of lifted arcs using the encoding scheme called HEAD by Nivre and Nilsson (2005), which means that a lifted arc is assigned the label $r{\uparrow}h$, where r is the original label and h is the label of the original head in the non-projective dependency graph.

Non-projective dependencies can be recovered by applying an inverse transformation to the output of the parser, using a left-to-right, top-down, breadth-first search, guided by the extended arc labels $r{\uparrow}h$ assigned by the parser. This technique has been used without exception for all languages.

3 Experiments

Since the projective parsing algorithm and graph transformation techniques are the same for all data sets, our optimization efforts have been focused on *feature selection*, using a combination of backward and forward selection starting from the base model described in section 2.2, and *parameter optimization* for the SVM learner, using grid search for an optimal combination of the kernel parameters γ and r, the penalty parameter C and the termination criterion ϵ, as well as the splitting feature s and the frequency threshold t. Feature selection and parameter optimization have to some extent been interleaved, but the amount of work done varies between languages.

[2] The fields PHEAD and PDEPREL have not been used at all, since we rely on pseudo-projective parsing for the treatment of non-projective structures.

[3] We also ran preliminary experiments with memory-based learning but found that this gave consistently lower accuracy.

[4] Preliminary experiments showed a slight improvement for most languages when splitting the FEATS values, as opposed to taking every combination of atomic values as a distinct value.

	Ara	Bul	Chi	Cze	Dan	Dut	Ger	Jap	Por	Slo	Spa	Swe	Tur	Total
LAS	66.71	87.41	86.92	78.42	84.77	78.59	85.82	91.65	87.60	70.30	81.29	84.58	65.68	80.19
UAS	77.52	91.72	90.54	84.80	89.80	81.35	88.76	93.10	91.22	78.72	84.67	89.50	75.82	85.48
LAcc	80.34	90.44	89.01	85.40	89.16	83.69	91.03	94.34	91.54	80.54	90.06	87.39	78.49	86.75

Table 2: Evaluation on final test set; LAS = labeled attachment score, UAS = unlabeled attachment score, LAcc = label accuracy score; total score excluding Bulgarian

The main optimization criterion has been labeled attachment score on held-out data, using ten-fold cross-validation for all data sets with 100k tokens or less, and an 80-20 split into training and devtest sets for larger datasets. The number of features in the optimized models varies from 16 (Turkish) to 30 (Spanish), but the models use all fields available for a given language, except that FORM is not used for Turkish (only LEMMA). The SVM parameters fall into the following ranges: γ: 0.12–0.20; r: 0.0–0.6; C: 0.1–0.7; ϵ: 0.01–1.0. Data has been split on the POS of the next input token for Czech (t = 200), German (t = 1000), and Spanish (t = 1000), and on the CPOS of the next input token for Bulgarian (t = 1000), Slovene (t = 600), and Turkish (t = 100). (For the remaining languages, the training data has not been split at all.)[5] A dry run at the end of the development phase gave a labeled attachment score of 80.46 over the twelve required languages.

Table 2 shows final test results for each language and for the twelve required languages together. The total score is only 0.27 percentage points below the score from the dry run, which seems to indicate that models have not been overfitted to the training data. The labeled attachment score varies from 91.65 to 65.68 but is above average for all languages. We have the best reported score for Japanese, Swedish and Turkish, and the score for Arabic, Danish, Dutch, Portuguese, Spanish, and overall does not differ significantly from the best one. The unlabeled score is less competitive, with only Turkish having the highest reported score, which indirectly indicates that the integration of labels into the parsing process primarily benefits labeled accuracy.

4 Error Analysis

An overall error analysis is beyond the scope of this paper, but we will offer a few general observations before we turn to Swedish and Turkish, focusing on recall and precision of root nodes, as a reflection of global syntactic structure, and on attachment score as a function of arc length. If we start by considering languages with a labeled attachment score of 85% or higher, they are characterized by high precision and recall for root nodes, typically 95/90, and by a graceful degradation of attachment score as arcs grow longer, typically 95–90–85, for arcs of length 1, 2 and 3–6. Typical examples are Bulgarian (Simov et al., 2005; Simov and Osenova, 2003), Chinese (Chen et al., 2003), Danish (Kromann, 2003), and Swedish (Nilsson et al., 2005). Japanese (Kawata and Bartels, 2000), despite a very high accuracy, is different in that attachment score drops from 98% to 85%, as we go from length 1 to 2, which may have something to do with the data consisting of transcribed speech with very short utterances.

A second observation is that a high proportion of non-projective structures leads to fragmentation in the parser output, reflected in lower precision for roots. This is noticeable for German (Brants et al., 2002) and Portuguese (Afonso et al., 2002), which still have high overall accuracy thanks to very high attachment scores, but much more conspicuous for Czech (Böhmová et al., 2003), Dutch (van der Beek et al., 2002) and Slovene (Džeroski et al., 2006), where root precision drops more drastically to about 69%, 71% and 41%, respectively, and root recall is also affected negatively. On the other hand, all three languages behave like high-accuracy languages with respect to attachment score. A very similar pattern is found for Spanish (Civit Torruella and Martí Antonín, 2002), although this cannot be explained by a high proportion of non-projective structures. One possible explanation in this case may be the fact that dependency graphs in the Spanish data are sparsely labeled, which may cause problem for a parser that relies on dependency labels as features.

The results for Arabic (Hajič et al., 2004; Smrž et al., 2002) are characterized by low root accuracy

[5] Detailed specifications of the feature models and learning algorithm parameters can be found on the MaltParser web page.

as well as a rapid degradation of attachment score with arc length (from about 93% for length 1 to 67% for length 2). By contrast, Turkish (Oflazer et al., 2003; Atalay et al., 2003) exhibits high root accuracy but consistently low attachment scores (about 88% for length 1 and 68% for length 2). It is noteworthy that Arabic and Turkish, being "typological outliers", show patterns that are different both from each other and from most of the other languages.

4.1 Swedish

A more fine-grained analysis of the Swedish results reveals a high accuracy for function words, which is compatible with previous studies (Nivre, 2006). Thus, the labeled F-score is 100% for infinitive markers (IM) and subordinating conjunctions (UK), and above 95% for determiners (DT). In addition, subjects (SS) have a score above 90%. In all these cases, the dependent has a configurationally defined (but not fixed) position with respect to its head.

Arguments of the verb, such as objects (DO, IO) and predicative complements (SP), have a slightly lower accuracy (about 85% labeled F-score), which is due to the fact that they "compete" in the same structural positions, whereas adverbials (labels that end in A) have even lower scores (often below 70%). The latter result must be related both to the relatively fine-grained inventory of dependency labels for adverbials and to attachment ambiguities that involve prepositional phrases. The importance of this kind of ambiguity is reflected also in the drastic difference in accuracy between noun pre-modifiers (AT) ($F > 97\%$) and noun post-modifiers (ET) ($F \approx 75\%$).

Finally, it is worth noting that coordination, which is often problematic in parsing, has high accuracy. The Swedish treebank annotation treats the second conjunct as a dependent of the first conjunct and as the head of the coordinator, which seems to facilitate parsing.[6] The attachment of the second conjunct to the first (CC) has a labeled F-score above 80%, while the attachment of the coordinator to the second conjunct (++) has a score well above 90%.

4.2 Turkish

In Turkish, very essential syntactic information is contained in the rich morphological structure, where concatenated suffixes carry information that in other languages may be expressed by separate words. The Turkish treebank therefore divides word forms into smaller units, called inflectional groups (IGs), and the task of the parser is to construct dependencies between IGs, not (primarily) between word forms (Eryiğit and Oflazer, 2006). It is then important to remember that an unlabeled attachment score of 75.8% corresponds to a word-to-word score of 82.7%, which puts Turkish on a par with languages like Czech, Dutch and Spanish. Moreover, when we break down the results according to whether the head of a dependency is part of a multiple-IG word or a complete (single-IG) word, we observe a highly significant difference in accuracy, with only 53.2% unlabeled attachment score for multiple-IG heads versus 83.7% for single-IG heads. It is hard to say at this stage whether this means that our methods are ill-suited for IG-based parsing, or whether it is mainly a case of sparse data for multiple-IG words.

When we break down the results by dependency type, we can distinguish three main groups. The first consists of determiners and particles, which have an unlabeled attachment score over 80% and which are found within a distance of 1–1.4 IGs from their head.[7] The second group mainly contains subjects, objects and different kinds of adjuncts, with a score in the range 60–80% and a distance of 1.8–5.2 IGs to their head. In this group, information about case and possessive features of nominals is important, which is found in the FEATS field in the data representation. We believe that one important explanation for our relatively good results for Turkish is that we break down the FEATS information into its atomic components, independently of POS and CPOS tags, and let the classifier decide which one to use in a given situation. The third group contains distant dependencies, such as sentence modifiers, vocatives and appositions, which have a much lower accuracy.

5 Conclusion

The evaluation shows that labeled pseudo-projective dependency parsing, using a deterministic parsing algorithm and SVM classifiers, gives competitive parsing accuracy for all languages involved in the

[6] The analysis is reminiscent of the treatment of coordination in the Collins parser (Collins, 1999).

[7] Given that the average IG count of a word is 1.26 in the treebank, this means that they are normally adjacent to the head word.

shared task, although the level of accuracy varies considerably between languages. To analyze in depth the factors determining this variation, and to improve our parsing methods accordingly to meet the challenges posed by the linguistic diversity, will be an important research goal for years to come.

Acknowledgments

We are grateful for the support from TÜBİTAK (The Scientific and Technical Research Council of Turkey) and the Swedish Research Council. We also want to thank Atanas Chanev for assistance with Slovene, the organizers of the shared task for all their hard work, and the creators of the treebanks for making the data available.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*. Kluwer Academic Publishers, Dordrecht.

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. "Floresta sintá(c)tica": a treebank for Portuguese. In *Proc. of LREC-2002*, pages 1698–1703.

N. B. Atalay, K. Oflazer, and B. Say. 2003. The annotation process in the Turkish treebank. In *Proc. of LINC-2003*.

E. Black, F. Jelinek, J. D. Lafferty, D. M. Magerman, R. L. Mercer, and S. Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proc. of the 5th DARPA Speech and Natural Language Workshop*, pages 31–37.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proc. of TLT-2002*.

C.-C. Chang and C.-J. Lin, 2001. *LIBSVM: A Library for Support Vector Machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

M. Civit Torruella and M^{a} A. Martí Antonín. 2002. Design principles for a Spanish treebank. In *Proc. of TLT-2002*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proc. of LREC-2006*.

G. Eryiğit and K. Oflazer. 2006. Statistical dependency parsing of Turkish. In *Proc. of EACL-2006*.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of NEMLAR-2004*, pages 110–117.

Y. Kawata and J. Bartels. 2000. Stylebook for the Japanese treebank in VERBMOBIL. Verbmobil-Report 240, Seminar für Sprachwissenschaft, Universität Tübingen.

M. T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In *Proc. of TLT-2003*.

T. Kudo and Y. Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proc. of CoNLL-2002*, pages 63–69.

J. Nilsson, J. Hall, and J. Nivre. 2005. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. In *Proc. of the NODALIDA Special Session on Treebanks*.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL-2005*, pages 99–106.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proc. CoNLL-2004*, pages 49–56.

J. Nivre, J. Hall, and J. Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proc. of LREC-2006*.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT-2003*, pages 149–160.

J. Nivre. 2006. *Inductive Dependency Parsing*. Springer.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15.

K. Simov and P. Osenova. 2003. Practical annotation scheme for an HPSG treebank of Bulgarian. In *Proc. of LINC-2003*, pages 17–24.

K. Simov, P. Osenova, A. Simov, and M. Kouylekov. 2005. Design and implementation of the Bulgarian HPSG-based treebank. In *Journal of Research on Language and Computation – Special Issue*, pages 495–522. Kluwer Academic Publishers.

O. Smrž, J. Šnaidauf, and P. Zemánek. 2002. Prague dependency treebank for Arabic: Multi-level annotation of Arabic corpus. In *Proc. of the Intern. Symposium on Processing of Arabic*, pages 147–155.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT-2003*, pages 195–206.

Multi-lingual Dependency Parsing with Incremental Integer Linear Programming

Sebastian Riedel and **Ruket Çakıcı** and **Ivan Meza-Ruiz**
ICCS
School of Informatics
University of Edinburgh
Edinburgh, EH8 9LW, UK
S.R.Riedel,R.Cakici,I.V.Meza-Ruiz@sms.ed.ac.uk

Abstract

Our approach to dependency parsing is based on the linear model of McDonald et al.(McDonald et al., 2005b). Instead of solving the linear model using the Maximum Spanning Tree algorithm we propose an incremental Integer Linear Programming formulation of the problem that allows us to enforce linguistic constraints. Our results show only marginal improvements over the non-constrained parser. In addition to the fact that many parses did not violate any constraints in the first place this can be attributed to three reasons: 1) the next best solution that fulfils the constraints yields equal or less accuracy, 2) noisy POS tags and 3) occasionally our inference algorithm was too slow and decoding timed out.

1 Introduction

This paper presents our submission for the CoNLL 2006 shared task of multilingual dependency parsing. Our parser is inspired by McDonald et al.(2005a) which treats the task as the search for the highest scoring Maximum Spanning Tree (MST) in a graph. This framework is efficient for both projective and non-projective parsing and provides an online learning algorithm which combined with a rich feature set creates state-of-the-art performance across multiple languages (McDonald and Pereira, 2006).

However, McDonald and Pereira (2006) mention the restrictive nature of this parsing algorithm. In their original framework, features are only defined over single attachment decisions. This leads to cases where basic linguistic constraints are not satisfied (e.g. verbs with two subjects). In this paper we present a novel way to implement the parsing algorithms for projective and non-projective parsing based on a more generic incremental Integer Linear Programming (ILP) approach. This allows us to include additional global constraints that can be used to impose linguistic information.

The rest of the paper is organised in the following way. First we give an overview of the Integer Linear Programming model and how we trained its parameters. We then describe our feature and constraint sets for the 12 different languages of the task (Hajič et al., 2004; Chen et al., 2003; Böhmová et al., 2003; Kromann, 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit Torruella and Martí Antonín, 2002; Nilsson et al., 2005; Oflazer et al., 2003; Atalay et al., 2003). Finally, our results are discussed and error analyses for Chinese and Turkish are presented.

2 Model

Our model is based on the linear model presented in McDonald et al. (2005a),

$$(1) \quad s(\mathbf{x}, \mathbf{y}) = \sum_{(i,j) \in \mathbf{y}} s(i,j) = \sum \mathbf{w} \cdot \mathbf{f}(i,j)$$

where $\mathbf{x}$ is a sentence, $\mathbf{y}$ a parse and s a score function over sentence-parse pairs. $\mathbf{f}(i,j)$ is a multidi-

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 226–230, New York City, June 2006.

mensional feature vector representation of the edge from token i to token j and $\mathbf{w}$ the corresponding weight vector. Decoding in this model amounts to finding the $\mathbf{y}$ for a given $\mathbf{x}$ that maximises $s(\mathbf{x}, \mathbf{y})$

$$y' = argmax_y s(\mathbf{x}, \mathbf{y})$$

and $\mathbf{y}$ contains no cycles, attaches exactly one head to each non-root token and no head to the root node.

2.1 Decoding

Instead of using the MST algorithm (McDonald et al., 2005b) to maximise equation 1, we present an equivalent ILP formulation of the problem. An advantage of a general purpose inference technique is the addition of further linguistically motivated constraints. For instance, we can add constraints that enforce that a verb can not have more than one subject argument or that coordination arguments should have compatible types. Roth and Yih (2005) is similarly motivated and uses ILP to deal with additional hard constraints in a Conditional Random Field model for Semantic Role Labelling.

There are several explicit formulations of the MST problem as integer programs in the literature (Williams, 2002). They are based on the concept of eliminating subtours (cycles), cuts (disconnections) or requiring intervertex flows (paths). However, in practice these cause long solving times. While the first two types yield an exponential number of constraints, the latter one scales cubically but produces non-fractional solutions in its relaxed version, causing long runtime of the branch and bound algorithm. In practice solving models of this form did not converge after hours even for small sentences.

To get around this problem we followed an incremental approach akin to Warme (1998). Instead of adding constraints that forbid all possible cycles in advance (this would result in an exponential number of constraints) we first solve the problem without any cycle constraints. Only if the result contains cycles we add constraints that forbid these cycles and run the solver again. This process is repeated until no more violated constraints are found. Figure 1 shows this algorithm.

Groetschel et al. (1981) showed that such an approach will converge after a polynomial number of iterations with respect to the number of variables.

1. Solve IP P_i
2. Find violated constraints C in the solution of P_i
3. if $C = \emptyset$ we are done
4. $P_{i+1} = P_i \cup C$
5. $i = i + 1$
6. goto (1)

Figure 1: Incremental Integer Linear Programming

In practice, this technique showed fast convergence (less than 10 iterations) in most cases, yielding solving times of less than 0.5 seconds. However, for some sentences in certain languages, such as Chinese or Swedish, an optimal solution could not be found after 500 iterations.

In the following section we present the bjective function, variables and linear constraints that make up the Integer Linear Program.

2.1.1 Variables

In the implementation[1] of McDonald et al. (2005b) dependency labels are handled by finding the best scoring label for a given token pair so that

$$s(i, j) = max\ s(i, j, label)$$

goes into Equation 1. This is only exact as long as no further constraints are added. Since our aim is to add constraints our variables need to explicitly model label decisions. Therefore, we introduce binary variables

$$l_{i,j,label} \forall i \in 0..n, j \in 1..n, label \in best_b(i, j)$$

where n is the number of tokens and the index 0 represents the root token. $best_b(i, j)$ is the set of b labels with maximal $s(i, j, label)$. $l_{i,j,label}$ equals 1 if there is a dependency with the label $label$ between token i (head) and j (child), 0 otherwise.

Furthermore, we introduce binary auxiliary variables

$$d_{i,j} \forall i \in 0..n, j \in 1..n$$

representing the existence of a dependency between tokens i and j. We connect these to the $l_{i,j,label}$ variables by a constraint

$$d_{i,j} = \sum_{label} l_{i,j,label}$$

.

[1] Note, however, that labelled parsing is not described in the publication.

2.1.2 Objective Function

Given the above variables our objective function can be represented as

$$\sum_{i,j} \sum_{label \in best_k(i,j)} s\left(i, j, label\right) \cdot l_{i,j,label}$$

with a suitable k.

2.1.3 Constraints Added in Advance

Only One Head In all our languages every token has exactly one head. This yields

$$\sum_{i>0} d_{i,j} = 1$$

for non-root tokens $j > 0$ and

$$\sum_{i} d_{i,0} = 0$$

for the artificial root node.

Typed Arity Constraints We might encounter solutions of the basic model that contain, for instance, verbs with two subjects. To forbid these we simply augment our model with constraints such as

$$\sum_{j} l_{i,j,subject} \leq 1$$

for all verbs i in a sentence.

2.1.4 Incremental Constraints

No Cycles If a solution contains one or more cycles C we add the following constraints to our IP: For every $c \in C$ we add

$$\sum_{(i,j) \in c} d_{i,j} \leq |c| - 1$$

to forbid c.

Coordination Argument Constraints In coordination conjuncts have to be of compatible types. For example, nouns can not coordinate with verbs. We implemented this constraint by checking the parses for occurrences of incompatible arguments. If we find two arguments j, k for a conjunction i: $d_{i,j}$ and $d_{i,k}$ and j is a noun and k is a verb then we add

$$d_{i,j} + d_{i,k} \leq 1$$

to forbid configurations in which both dependencies are active.

Projective Parsing In the incremental ILP framework projective parsing can be easily implemented by checking for crossing dependencies after each iteration and forbidding them in the next. If we see two dependencies that cross, $d_{i,j}$ and $d_{k,l}$, we add the constraint

$$d_{i,j} + d_{k,l} \leq 1$$

to prevent this in the next iteration. This can also be used to prevent specific types of crossings. For instance, in Dutch we could only allow crossing dependencies as long as none of the dependencies is a "Determiner" relation.

2.2 Training

We used single-best MIRA(Crammer and Singer, 2003).For all experiments we used 10 training iterations and non-projective decoding. Note that we used the original spanning tree algorithm for decoding during training as it was faster.

3 System Summary

We use four different feature sets. The first feature set, BASELINE, is taken from McDonald and Pereira (2005b). It uses the *FORM* and the *POSTAG* fields. This set also includes features that combine the label and POS tag of head and child such as $(Label, POS_{Head})$ and $(Label, POS_{Child-1})$. For our Arabic and Japanese development sets we obtained the best results with this configuration. We also use this configuration for Chinese, German and Portuguese because training with other configurations took too much time (more than 7 days).

The BASELINE also uses pseudo-coarse-POS tag ($1st$ character of the *POSTAG*) and pseudo-lemma tag (4 characters of the *FORM* when the length is more than 3). For the next configuration we substitute these pseudo-tags by the *CPOSTAG* and *LEMMA* fields that were given in the data. This configuration was used for Czech because for other configurations training could not be finished in time.

The third feature set tries to exploit the generic *FEATS* field, which can contain a list features such as case and gender. A set of features per dependency is extracted using this information. It consists of cross product of the features in *FEATS*. We used this configuration for Danish, Dutch, Spanish

and Turkish where it showed the best results during development.

The fourth feature set uses the triplet of label, *POS* child and head as a feature such as $(Label, POS_{Head}, POS_{Child})$. It also uses the *CPOSTAG* and *LEMMA* fields for the head. This configuration is used for Slovene and Swedish data where it performed best during development.

Finally, we add constraints for Chinese, Dutch, Japanese and Slovene. In particular, arity constraints to Chinese and Slovene, coordination and arity constraints to Dutch, arity and selective projectivity constraints for Japanese[2]. For all experiments b was set to 2. We did not apply additional constraints to any other languages due to lack of time.

4 Results

Our results on the test set are shown in Table 1. Our results are well above the average for all languages but Czech. For Chinese we perform significantly better than all other participants ($p = 0.00$) and we are in the top three entries for Dutch, German, Danish. Although Dutch and Chinese are languages were we included additional constraints, our scores are not a result of these. Table 2 compares the result for the languages with additional constraints. Adding constraints only marginally helps to improve the system (in the case of Slovene a bug in our implementation even degraded accuracy). A more detailed explanation to this observation is given in the following section. A possible explanation for our high accuracy in Chinese could be the fact that we were not able to optimise the feature set on the development set (see the previous section). Maybe this prevented us from overfitting. It should be noted that we did use non-projective parsing for Chinese, although the corpus was fully projective. Our worst results in comparison with other participants can be seen for Czech. We attribute this to the reduced training set we had to use in order to produce a model in time, even when using the original MST algorithm.

[2]This is done in order to capture the fact that crossing dependencies in Japanese could only be introduced through disfluencies.

4.1 Chinese

For Chinese the parser was augmented with a set of constraints that disallowed more than one argument of the types *head, goal, nominal, range, theme, reason, DUMMY, DUMMY1* and *DUMMY2*.

By enforcing arity constraints we could either turn wrong labels/heads into right ones and improve accuracy or turn right labels/heads into wrong ones and degrade accuracy. For the test set the number of improvements (36) was higher than the number of errors (22). However, this margin was outweighed by a few sentences we could not properly process because our inference method timed out. Our overall improvement was thus unimpressive 7 tokens.

In the context of duplicate "head" dependencies (that is, dependencies labelled "head") the number of sentences where accuracy dropped far outweighed the number of sentences where improvements could be gained. Removing the arity constraints on "head" labels therefore should improve our results.

This shows the importance of good second best dependencies. If the dependency with the second highest score is the actual gold dependency and its score is close to the highest score, we are likely to pick this dependency in the presence of additional constraints. On the other hand, if the dependency with the second highest score is not the gold one and its score is too high, we will probably include this dependency in order to fulfil the constraints.

There may be some further improvement to be gained if we train our model using k-best MIRA with $k > 1$ since it optimises weights with respect to the k best parses.

4.2 Turkish

There is a considerable gap between the unlabelled and labelled results for Turkish. And in terms of labels the POS type *Noun* gives the worst performance because many times a subject was classified as object or vice a versa.

Case information in Turkish assigns argument roles for nouns by marking different semantic roles. Many errors in the Turkish data might have been caused by the fact that this information was not adequately used. Instead of fine-tuning our feature set to Turkish we used the feature cross product as de-

Model	AR	CH	CZ	DA	DU	GE	JP	PO	SL	SP	SW	TU
OURS	66.65	89.96	67.64	83.63	78.59	86.24	90.51	84.43	71.20	77.38	80.66	58.61
AVG	59.94	78.32	67.17	78.31	70.73	78.58	85.86	80.63	65.16	73.53	76.44	55.95
TOP	66.91	89.96	80.18	84.79	79.19	87.34	91.65	87.60	73.44	82.25	84.58	65.68

Table 1: Labelled accuracy on the test sets.

Constraints	DU	CH	SL	JA
with	3927	4464	3612	4526
without	3928	4471	3563	4528

Table 2: Number of tokens correctly classified with and without constraints.

scribed in Section 3. Some of the rather meaningless combinations might have neutralised the effect of sensible ones. We believe that using morphological case information in a sound way would improve both the unlabelled and the labelled dependencies. However, we have not performed a separate experiment to test if using the case information alone would improve the system any better. This could be the focus of future work.

5 Conclusion

In this work we presented a novel way of solving the linear model of McDonald et al. (2005a) for projective and non-projective parsing based on an incremental ILP approach. This allowed us to include additional linguistics constraints such as "a verb can only have one subject."

Due to time constraints we applied additional constraints to only four languages. For each one we gained better results than the baseline without constraints, however, this improvement was only marginal. This can be attributed to 4 main reasons: Firstly, the next best solution that fulfils the constraints was even worse (Chinese). Secondly, noisy POS tags caused coordination constraints to fail (Dutch). Thirdly, inference timed out (Chinese) and fourthly, constraints were not violated that often in the first place (Japanese).

However, the effect of the first problem might be reduced by training with a higher k. The second problem could partly be overcome by using a better tagger or by a special treatment within the constraint handling for word types which are likely to be mistagged. The third problem could be avoidable by adding constraints during the branch and bound algorithm, avoiding the need to resolve the full problem "from scratch" for every constraint added. With these remedies significant improvements to the accuracy for some languages might be possible.

6 Acknowledgements

We would like to thank Beata Kouchnir, Abhishek Arun and James Clarke for their help during the course of this project.

References

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991.

M. Groetschel, L. Lovasz, and A. Schrijver. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, I:169– 197.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of the 11th Annual Meeting of the EACL*.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. of the 43rd Annual Meeting of the ACL*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP 2005*, Vancouver, B.C., Canada.

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 737–744.

David Michael Warme. 1998. *Spanning Trees in Hypergraphs with Application to Steiner Trees*. Ph.D. thesis, University of Virginia.

Justin C. Williams. 2002. A linear-size zero - one programming model for the minimum spanning tree problem in planar graphs. *Networks*, 39:53–60.

Language Independent Probabilistic Context-Free Parsing Bolstered by Machine Learning

Michael Schiehlen Kristina Spranger
Institute for Computational Linguistics
University of Stuttgart
D-70174 Stuttgart
Michael.Schiehlen@ims.uni-stuttgart.de
Kristina.Spranger@ims.uni-stuttgart.de

Abstract

Unlexicalized probabilistic context-free parsing is a general and flexible approach that sometimes reaches competitive results in multilingual dependency parsing even if a minimum of language-specific information is supplied. Furthermore, integrating parser results (good at long dependencies) and tagger results (good at short range dependencies, and more easily adaptable to treebank peculiarities) gives competitive results in all languages.

1 Introduction

Unlexicalized probabilistic context-free parsing is a simple and flexible approach that nevertheless has shown good performance (Klein and Manning, 2003). We applied this approach to the shared task (Buchholz et al., 2006) for Arabic (Hajič et al., 2004), Chinese (Chen et al., 2003), Czech (Böhmová et al., 2003), Danish (Kromann, 2003), Dutch (van der Beek et al., 2002), German (Brants et al., 2002), Japanese (Kawata and Bartels, 2000), Portuguese (Afonso et al., 2002), Slovene (Džeroski et al., 2006), Spanish (Civit Torruella and Martí Antonín, 2002), Swedish (Nilsson et al., 2005), Turkish (Oflazer et al., 2003; Atalay et al., 2003), but not Bulgarian (Simov et al., 2005). In our approach we put special emphasis on language independence: We did not use any extraneous knowledge; we did not do any transformations on the treebanks; we restricted language-specific parameters to a small, easily manageable set (a classification of dependency relations into complements, adjuncts, and conjuncts/coordinators, and a switch for Japanese to include coarse POS tag information, see section 3.4). In a series of post-submission experiments, we investigated how much the parse results can help a machine learner.

2 Experimental Setup

For development, we chose the initial n sentences of every treebank, where n is the number of the sentences in the test set. In this way, the sizes were realistic for the task. For parsing the test data, we added the development set to the training set.

All the evaluations on the test sets were performed with the evaluation script supplied by the conference organizers. For development, we used labelled F-score computed from all tokens except the ones employed for punctuation (cf. section 3.2).

3 Context Free Parsing

3.1 The Parser

Basically, we investigated the performance of a straightforward unlexicalized statistical parser, viz. BitPar (Schmid, 2004). BitPar is a CKY parser that uses bit vectors for efficient representation of the chart and its items. If frequencies for the grammatical and lexical rules in a training set are available, BitPar uses the Viterbi algorithm to extract the most probable parse tree (according to PCFG) from the chart.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 231–235, New York City, June 2006.

3.2 Converting Dependency Structure to Constituency Structure

In order to determine the grammar rules required by the context-free parser, the dependency trees in the CONLL format have to be converted to constituency trees. Gaifman (1965) proved that projective dependency grammars can be mapped to context-free grammars. The main information that needs to be added in going from dependency to constituency structure is the category of non-terminals. The usage of special knowledge bases to determine projections of categories (Xia and Palmer, 2001) would have presupposed language-dependent knowledge, so we investigated two other options: Flat rules (Collins et al., 1999) and binary rules. In the flat rules approach, each lexical category projects to exactly one phrasal category, and every projection chain has a length of at most one. The binary rules approach makes use of the X-bar-scheme and thus introduces along with the phrasal category an intermediate category. The phrasal category must not occur more than once in a projection chain, and a projection chain must not end in an intermediate category. In both approaches, projection is only triggered if dependents are present; in case a category occurs as a dependent itself, no projection is required. In coordination structures, the parent category is copied from that of the last conjunct.

Non-projective relations can be treated as unbounded dependencies so that their surface position (antecedent position) is related to the position of their head (trace position) with an explicit coindexed trace (like in the Penn treebank). To find the position of trace and antecedent we assume three constraints: The antecedent should c-command its trace. The antecedent is maximally near to the trace in depth of embedding. The trace is maximally near to the antecedent in surface order.

Finally the placement of punctuation signs has a major impact on the performance of a parser (Collins et al., 1999). In most of the treebanks, not much effort is invested into the treatment of punctuation. Sometimes, punctuation signs play a role in predicate-argument structure (commas acting as coordinators), but more often they do not, in which case they are marked by special roles (e.g. “pnct”, “punct”, “PUNC”, or “PUNCT”). We used a general mechanism to re-insert such signs, for all languages but CH (no punctuation signs) and AR, CZ, SL (reliable annotation). Correct placement of punctuation presupposes knowledge of the punctuation rules valid in a language. In the interest of generality, we opted for a suboptimal solution: Punctuation signs are inserted in the highest possible position in a tree.

3.3 Subcategorization and Coordination

The most important language-specific information that we made use of was a classification of dependency relations into complements, coordinators/conjuncts, and other relations (adjuncts).

Given knowledge about complement relations, it is fairly easy to construct subcategorization frames for word occurrences: A subcategorization frame is simply the set of the complement relations by which dependents are attached to the word. To give the parser access to these lists, we annotated the category of a subcategorizing word with its subcategorization frame. In this way, the parser can learn to associate the subcategorization requirements of a word with its local syntactic context (Schiehlen, 2004).

Coordination constructions are marked either in the conjuncts (CH, CZ, DA, DU, GE, PO, SW) or the coordinator (AR, SL). If conjuncts show coordination, a common representation of asyndetic coordination has one conjunct point to another conjunct. It is therefore important to distinguish coordinators from conjuncts. Coordinators are either singled out by special dependency relations (DA, PO, SW) or by their POS tags (CH, DU). In German, the first conjunct phrase is merged with the whole coordinated phrase (due to a conversion error?) so that determining the coordinator as a head is not possible.

We also experimented with attaching the POS tags of heads to the categories of their adjunct dependents. In this way, the parser could differentiate between e.g. verbal and nominal adjuncts. In our experiments, the performance gains achieved by this strategy were low, so we did not incorporate it into the system. Possibly, better results could be achieved by restricting annotation to special classes of adjuncts or by generalizing the heads’ POS tags.

3.4 Categories

As the treebanks provide a lot of information with every word token, it is a delicate question to de-

	Ch	Da	Du	Ge	Ja	Po	Sp	Tu
coarse POS	72.99	69.38	69.27	–	79.07		66.09	
fine POS	61.21	69.78	67.72	7.40	73.44	71.75		54.96
POS + feat	–	42.67	40.40	–				
dep-rel	76.61	72.77	70.70	70.31	78.12	72.93	66.93	65.03
coarse + dep-rel	77.61	67.56	69.43	–	81.36		64.03	
fine + dep-rel	51.21	57.72	68.55			46.28	36.59	54.97

Figure 1: Types of Categories (Development Results)

cide on the type and granularity of the information to use in the categories of the grammar. The treebanks specify for every word a (fine-grained) POS tag, a coarse-grained POS tag, a collection of morphosyntactic features, and a dependency relation (dep-rel). Only the dependency relation is really orthogonal; the other slots contain various generalizations of the same morphological information. We tested several options: coarse-grained POS tag (if available), fine-grained POS tag, fine-grained POS tag with morphosyntactic features (if available), name of dependency relation, and the combinations of coarse-grained or fine-grained POS tags with the dependency relation.

Figure 1 shows F-score results on the development set for several languages and different combinations. The best overall performer is dep-rel; this somewhat astonishing fact may be due to the superior quality of the annotations in this slot (dependency relations were annotated by hand, POS tags automatically). Furthermore, being checked in evaluation, dependency relations directly affect performance. Since we wanted a general language-independent strategy, we used always the dep-rel tags but for Japanese. The Japanese treebank features only 8 different dependency relations, so we added coarse-grained POS tag information. In the categories for Czech, we deleted the suffixes marking coordination, apposition and parenthesis (Co, Ap, Pa), reducing the number of categories roughly by a factor of four. In coordination, conjuncts inherit the dep-rel category from the parent.

Whereas the dep-rel information is submitted to the parser directly in terms of the categories, the information in the lemma, POS tag and morphosyntactic features slot was used only for back-off smoothing when associating lexical items with categories. A grammar with this configuration was used to produce the results submitted (cf. line labelled CF in Figures 2 and 3).

	Cz	Ge	Sp	Sw
dep-rel	52.66	70.31	66.93	72.91
new classific	58.92	74.32	66.09	61.59
new + dep-rel	56.94	78.40	64.03	66.32

Figure 4: Manual POS Tag Classes (Development)

Instead of using the category generalizations supplied with the treebanks directly, manual labour can be put into discovering classifications that behave better for the purposes of statistical parsing. So, Collins et al. (1999) proposed a tag classification for parsing the Czech treebank. We also investigated a classification for German[1], as well as one for Swedish and one for Spanish, which were modelled after the German classification. The results in Figure 4 show that new classifications may have a dramatic effect on performance if the treebank is sufficiently large. In the interest of generality, we did not make use of the language dependent tag classifications for the results submitted, but we will nevertheless report results that could have been achieved with these classifications.

3.5 Markovization

Another strategy that is often used in statistical parsing is Markovization (Collins, 1999): Treebanks

[1] punctuation {$($" $, $.} adjectives {ADJA ADJD CARD} adverbs {ADV PROAV PTKA PTKNEG PTKVZ PWAV} prepositions {APPR APPO APZR APPRART KOKOM} nouns {NN NE NNE PDS PIS PPER PPOSS PRELS PRF PWS SYM} determiners {ART PDAT PIAT PRELAT PPOSAT PWAT} verb forms {VAFIN VMFIN VVFIN} {VAIMP VVIMP} {VAINF VMINF VVINF} {VAPP VMPP VVPP} {VVIZU PTKZU} clause-like items {ITJ PTKANT KOUS}

	Ar	Ch	Cz	Da	Du	Ge	Ja	Po	Sl	Sp	Sw	Tu	Bu
Best	66.91	89.96	80.18	84.79	79.19	87.34	91.65	87.60	73.44	82.25	84.58	65.68	87.57
Average	59.94	78.32	67.17	76.16	70.73	78.58	85.86	80.63	65.16	73.52	76.44	55.95	79.98
CF (submitted)	44.39	66.20	53.34	76.05	72.11	68.73	83.35	71.01	50.72	46.96	71.10	49.81	–
MaxEnt	59.16	61.65	63.28	73.25	64.47	73.94	82.79	80.30	66.27	69.73	72.99	47.16	–
combined	**61.82**	73.34	71.74	**79.64**	**75.51**	80.75	**88.15**	**82.43**	67.09	71.15	**76.88**	**53.65**	–
CF+Markov	45.37	70.76	55.14	74.49	72.55	68.87	84.57	71.89	55.16	47.95	71.18	51.64	–
CFM+newcl		73.84	62.10			77.76				49.61			–
combined		**76.84**	**72.76**			**82.59**			**69.38**	**72.57**			–
new rules (in %)	7.15	6.03	4.64	7.34	5.03	7.42	5.59	6.69	21.00	9.50	10.14	14.23	

Figure 2: Labelled Accuracy Results on the Test Sets

	Ar	Ch	Cz	Da	Du	Ge	Ja	Po	Sl	Sp	Sw	Tu
CF	41.91	76.61	52.66	72.77	70.69	70.31	81.36	72.76	49.00	66.93	72.91	65.03
CF+Markov	63.00	80.25	52.80	73.31	70.70	70.51	82.59	74.37	52.43	67.81	73.56	82.80
CFM+newcl		83.07	59.03			80.42				69.30		

Figure 3: F Score Results on the Development Sets

usually contain very many long rules of low frequency (presumably because inserting nodes costs annotators time). Such rules cannot have an impact in a statistical system (the line new-rules in Figure 2 shows the percentage of rules in the test set that are not in the training set); it is better to view them as products of a Markov process that chooses first the head, then the symbols left of the head and finally the symbols right of the hand. In a bigram model, the choice of left and right siblings is made dependent not only on the parent and head category, but also on the last sibling on the left or right, respectively. Formally the probability of a rule with left hand side C and right hand side $L_n \ldots L_1 H R_1 \ldots R_m$ is broken down to the product of the probability $P_h(H|C)$ of the head, the probabilities of the left siblings $P_l(L_i|L_{i-1}, H, C)$ and those of the right siblings $P_r(R_i|R_{i-1}, H, C)$. Generic symbols designate beginning (L_0, R_0) and end (L_{n+1}, R_{m+1}) of the sibling lists. The method can be transferred to plain unlexicalized PCFG (Klein and Manning, 2003) by transforming long rules into a series of binary rules:

$$
\begin{aligned}
&C \leftarrow L_n \ \langle C, H, L_n, L_{n-1}\rangle \\
&\langle C, H, L_{i+1}, L_i\rangle \leftarrow L_i \ \langle C, H, L_i, L_{i-1}\rangle \\
&\langle C, H, L_1, L_0\rangle \leftarrow [C, H, R_n, R_{n-1}] \ R_n \\
&[C, H, R_{i+1}, R_i] \leftarrow [C, H, R_i, R_{i-1}] \ R_i \\
&[C, H, R_1, R_0] \leftarrow H
\end{aligned}
$$

If the bigram symbols $[C, H, R_i, R_{i-1}]$ and $\langle C, H, L_i, L_{i-1}\rangle$ occur in less than a certain number of rules (50 in our case), we smooth to unigram symbols instead ($[C, H, R_i]$ and $\langle C, H, L_i\rangle$). We used a script of Schmid (2006) to Markovize infrequent rules in this manner (i.e. all rules with less than 50 occurrences that are not coordination rules).

For time reasons, Markovization was not taken into account in the submitted results. We refer to Figures 2 and 3 (line labelled CF+Markov) for a listing of the results attainable by Markovization on the individual treebanks. Performance gains are even more dramatic if in addition dependency relations + manual POS tag classes are used as categories (line labelled CFM+newcl in Figures 2 and 3).

3.6 From Constituency Structure Back to Dependency Structure

In a last step, we converted the constituent trees back to dependency trees, using the algorithm of Gaifman (1965). Special provisos were necessary for the root node, for which no head is given in certain treebanks (Džeroski et al., 2006). To interpret the context-free rules, we associated their children with dependency relations. This information was kept in a separate file that was invisible to the parser. In cases there were several possible interpretations for a context

free rule, we always chose the most frequent one in the training data (Schiehlen, 2004).

4 Machine Learning

While the results coming from the statistical parser are not really competitive, we believe that they nevertheless present valuable information for a machine learner. To give some substance to this claim, we undertook experiments with the Zhang Le's MaxEnt Toolkit[2]. For this work, we recast the dependency parsing problem as a classification problem: Given some feature information on the word token, in which dependency relations does it stand to which head? While the representation of dependency relations is straightforward, the representation of heads is more difficult. Building on past experiments (Schiehlen, 2003), we chose the "nth-tag" representation which consists of three pieces of information: the POS tag of the head, the direction in which the head lies (left or right), and the number of words with the same POS tag between head and dependent. We used the following features to describe a word token: the fine-grained POS tag, the lemma (or full form) if it occurs at least 10 times, the morphosyntactic features, and the POS tags of the four preceding and the four following word tokens. The learner was trained in standard configuration (30 iterations). The results for this method on the test data are shown in Figure 2 (line MaxEnt).

In a second experiment we added parsing results (obtained by 10-fold cross validation on the training set) in two features: proposed dependency relation and proposed head. Results of the extended learning approach are shown in Figure 2 (line combined).

5 Conclusion

We have presented a general approach to parsing arbitrary languages based on dependency treebanks that uses a minimum overhead of language-specific information and nevertheless supplies competitive results in some languages (Da, Du). Even better results can be reached if POS tag classifications are used in the categories that are optimized for specific languages (Ge). Markovization usually brings an improvement of up to 2%, a higher gain is reached in Slovene (where many new rules occur in the testset) and Chinese (which has the highest number of dependency relations). Comparable results in the literature are Schiehlen's (2004) 81.03% dependency f-score reached on the German NEGRA treebank and Collins et al.'s (1999) 80.0% labelled accuracy on the Czech PDT treebank. Collins (1999) used a lexicalized approach, Schiehlen (2004) used the manually annotated phrasal categories of the treebank.

Our second result is that context-free parsing can also boost the performance of a simple tagger-like machine learning system. While a maximum-entropy learner on its own achieves competitive results for only three languages (Ar, Po, Sl), competitive results in basically all languages are produced with access to the results of the probabilistic parser.

Thanks go to Helmut Schmid for providing support with his parser and the Markovization script.

[2] http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

References

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL-X*. SIGNLL.

Michael J. Collins, Jan Hajiĉ, Lance Ramshaw, and Christoph Tillmann. 1999. A Statistical Parser for Czech. In *ACL'99*, College Park, MA.

Michael J. Collins. 1999. *Head-Driven Statistical Methods for Natural Language Parsing*. Ph.D. thesis, Univ. of Pennsylvania.

Haim Gaifman. 1965. Dependency Systems and Phrase-Structure Systems. *Information and Control*, 8(3):304–337.

Dan Klein and Christopher Manning. 2003. Accurate Unlexicalized Parsing. In *ACL'03*, pages 423–430.

Michael Schiehlen. 2003. Combining Deep and Shallow Approaches in Parsing German. In *ACL'03*, pages 112–119, Sapporo, Japan.

Michael Schiehlen. 2004. Annotation Strategies for Probabilistic Parsing in German. In *COLING '04*, pages 390–396, Geneva, Switzerland, August.

Helmut Schmid. 2004. Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit Vectors. In *COLING '04*, Geneva, Switzerland.

Helmut Schmid. 2006. Trace Prediction and Recovery with Unlexicalized PCFGs and Gap Threading. Submitted to *COLING '06*.

Fei Xia and Martha Palmer. 2001. Converting dependency structures to phrase structures. In *HLT 2001*.

Maximum Spanning Tree Algorithm for Non-projective Labeled Dependency Parsing

Nobuyuki Shimizu
Dept. of Computer Science
State University of New York at Albany
Albany, NY, 12222, USA
shimizu@cs.albany.edu

Abstract

Following (McDonald et al., 2005), we present an application of a maximum spanning tree algorithm for a directed graph to non-projective labeled dependency parsing. Using a variant of the voted perceptron (Collins, 2002; Collins and Roark, 2004; Crammer and Singer, 2003), we discriminatively trained our parser in an on-line fashion. After just one epoch of training, we were generally able to attain average results in the CoNLL 2006 Shared Task.

1 Introduction

Recently, we have seen dependency parsing grow more popular. It is not rare to see dependency relations used as features, in tasks such as relation extraction (Bunescu and Mooney, 2005) and machine translation (Ding and Palmer, 2005). Although English dependency relations are mostly projective, in other languages with more flexible word order, such as Czech, non-projective dependencies are more frequent. There are generally two methods for learning non-projective dependencies. You could map a non-projective dependency tree to a projective one, learn and predict the tree, then bring it back to the non-projective dependency tree (Nivre and Nilsson, 2005). Non-projective dependency parsing can also be represented as search for a maximum spanning tree in a directed graph, and this technique has been shown to perform well in Czech (McDonald et al., 2005). In this paper, we investigate the effectiveness of (McDonald et al., 2005) in the various languages given by the CoNLL 2006 shared task for non-projective labeled dependency parsing.

The paper is structured as follows: in section 2 and 3, we review the decoding and learning aspects of (McDonald et al., 2005), and in section 4, we describe the extension of the algorithm and the features needed for the CoNLL 2006 shared task.

2 Non-Projective Dependency Parsing

2.1 Dependency Structure

Let us define x to be a generic sequence of input tokens together with their POS tags and other morphological features, and y to be a generic dependency structure, that is, a set of edges for x. We use the terminology in (Taskar et al., 2004) for a generic structured output prediction, and define a *part*.

A *part* represents an edge together with its label. A part is a tuple $\langle DEPREL, i, j \rangle$ where i is the start point of the edge, j is the end point, and *DEPREL* is the label of the edge. The token at i is the head of the token at j.

Table 1 shows our formulation of building a non-projective dependency tree as a prediction problem. The task is to predict y, the set of parts (column 3, Table 1), given x, the input tokens and their features (column 1 and 2, Table 1).

In this paper we use the common method of factoring the score of the dependency structure as the sum of the scores of all the parts.

A dependency structure is characterized by its features, and for each feature, we have a correspond-

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 236–240, New York City, June 2006.

Token	POS	Edge Part
John	NN	$\langle SUBJ, 2, 1\rangle$
saw	VBD	$\langle PRED, 0, 2\rangle$
a	DT	$\langle DET, 4, 3\rangle$
dog	NN	$\langle OBJ, 2, 4\rangle$
yesterday	RB	$\langle ADJU, 2, 5\rangle$
which	WDT	$\langle MODWH, 7, 6\rangle$
was	VBD	$\langle MODPRED, 4, 7\rangle$
a	DT	$\langle DET, 10, 8\rangle$
Yorkshire	NN	$\langle MODN, 10, 9\rangle$
Terrier	NN	$\langle OBJ, 7, 10\rangle$
.	.	$\langle ., 10, 11\rangle$

Table 1: Example Parts

ing weight. The score of a dependency structure is the sum of these weights. Now, the dependency structures are factored by the parts, so that each feature is some type of a specialization of a part. Each part in a dependency structure maps to several features. If we sum up the weights for these features, we have the score for the part, and if we sum up the scores of the parts, we have the score for the dependency structure.

For example, let us say we would like to find the score of the part $\langle OBJ, 2, 4\rangle$. This is the edge going to the 4th token "dog" in Table 1. Suppose there are two features for this part.

- There is an edge labeled with "OBJ" that points to the right. (= *DEPREL*, $dir(i, j)$)
- There is an edge labeled with "OBJ" starting at the token "saw" which points to the right. (= *DEPREL*, $dir(i, j)$, $word_i$)

If a statement is never true during the training, the weight for it will be 0. Otherwise there will be a positive weight value. The score will be the sum of all the weights of the features given by the part.

In the upcoming section, we explain a decoding algorithm for the dependency structures, and later we give a method for learning the weight vector used in the decoding.

2.2 Maximum Spanning Tree Algorithm

As in (McDonald et al., 2005), the decoding algorithm we used is the Chu-Liu-Edmonds (CLE) algorithm (Chu and Liu, 1965; Edmonds, 1967) for finding the Maximum Spanning Tree in a directed graph. The following is a nice summary by (McDonald et al., 2005).

> Informally, the algorithm has each vertex in the graph greedily select the incoming edge with highest weight.

Note that the edge is coming from the parent to the child. This means that given a child node $word_j$, we are finding the parent, or the head $word_i$ such that the edge (i, j) has the highest weight among all i, $i \neq j$.

> If a tree results, then this must be the maximum spanning tree. If not, there must be a cycle. The procedure identifies a cycle and contracts it into a single vertex and recalculates edge weights going into and out of the cycle. It can be shown that a maximum spanning tree on the contracted graph is equivalent to a maximum spanning tree in the original graph (Leonidas, 2003). Hence the algorithm can recursively call itself on the new graph.

3 Online Learning

Again following (McDonald et al., 2005), we have used the single best MIRA (Crammer and Singer, 2003), which is a variant of the voted perceptron (Collins, 2002; Collins and Roark, 2004) for structured prediction. In short, the update is executed when the decoder fails to predict the correct parse, and we compare the correct parse y^t and the incorrect parse y' suggested by the decoding algorithm. The weights of the features in y' will be lowered, and the weights of the features in y^t will be increased accordingly.

4 Experiments

Our experiments were conducted on CoNLL-X shared task, with various datasets (Hajič et al., 2004; Simov et al., 2005; Simov and Osenova, 2003; Chen et al., 2003; Böhmová et al., 2003; Kromann, 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit Torruella and Martí Antonín, 2002; Nilsson et al., 2005; Oflazer et al., 2003; Atalay et al., 2003) .

4.1 Dependency Relation

The CLE algorithm works on a directed graph with unlabeled edges. Since the CoNLL-X shared task

Given a part $\langle DEPREL, i, j \rangle$
DEPREL, $dir(i,j)$
DEPREL, $dir(i,j)$, $word_i$
DEPREL, $dir(i,j)$, pos_i
DEPREL, $dir(i,j)$, $word_j$
DEPREL, $dir(i,j)$, pos_j
DEPREL, $dir(i,j)$, $word_i$, pos_i
DEPREL, $dir(i,j)$, $word_j$, pos_j
DEPREL, $dir(i,j)$, $word_{i-1}$
DEPREL, $dir(i,j)$, pos_{i-1}
DEPREL, $dir(i,j)$, $word_{i-1}$, pos_{i-1}
DEPREL, $dir(i,j)$, $word_{j-1}$
DEPREL, $dir(i,j)$, pos_{j-1}
DEPREL, $dir(i,j)$, $word_{j-1}$, pos_{j-1}
DEPREL, $dir(i,j)$, $word_{i+1}$
DEPREL, $dir(i,j)$, pos_{i+1}
DEPREL, $dir(i,j)$, $word_{i+1}$, pos_{i+1}
DEPREL, $dir(i,j)$, $word_{j+1}$
DEPREL, $dir(i,j)$, pos_{j+1}
DEPREL, $dir(i,j)$, $word_{j+1}$, pos_{j+1}
DEPREL, $dir(i,j)$, pos_{i-2}
DEPREL, $dir(i,j)$, pos_{i+2}
DEPREL, $dir(i,j)$, distance = $\lvert j-i \rvert$
additional features
DEPREL, $dir(i,j)$, $word_i$, $word_j$
DEPREL, $dir(i,j)$, pos_{i+1}, pos_i, pos_{i+1}
DEPREL, $dir(i,j)$, pos_{i+1}, $word_i$, pos_{i+1}
DEPREL, $dir(i,j)$, $word_i$, pos_i, pos_j
DEPREL, $dir(i,j)$, pos_i, $word_j$, pos_j

Table 2: Binary Features for Each Part

requires the labeling of edges, as a preprocessing stage, we created a directed complete graph without multi-edges, that is, given two distinct nodes i and j, exactly two edges exist between them, one from i to j, and the other from j to i. There is no self-pointing edge. Then we labeled each edge with the highest scoring dependency relation. This complete graph was given to the CLE algorithm and the edge labels were never altered in the course of finding the maximum spanning tree. The result is the non-projective dependency tree with labeled edges.

4.2 Features

The features we used to score each part (edge) $\langle DEPREL, i, j \rangle$ are shown in Table 2. The index i is the position of the parent and j is that of the child.

$word_j$ = the word token at the position j.

pos_j = the coarse part-of-speech at j.

$dir(i,j)$ = R if $i < j$, and L otherwise.

No other features were used beyond the combinations of the CPOS tag and the word token in Table 2.

We have evaluated our parser on Arabic, Danish, Slovene, Spanish, Turkish and Swedish, and used the "additional features" listed in Table 2 for all languages except for Danish and Swedish. The reason for this is simply that the model with the additional features did not fit in the 4 GB of memory used in the training.

Although we could do batch learning by running the online algorithm multiple times, we run the online algorithm just once. The hardware used is an Intel Pentinum D at 3.0 Ghz with 4 GB of memory, and the software was written in C++. The training time required was Arabic 204 min, Slovene 87 min, Spanish 413 min, Swedish 1192 min, Turkish 410 min, Danish 381 min.

5 Results

The results are shown in Table 3. Although our feature set is very simple, the results were around the averages. We will do error analysis of three notable languages: Arabic, Swedish and Turkish.

5.1 Arabic

Of 4990 words in the test set, 800 are prepositions. The prepositions are the most frequently found tokens after nouns in this set. On the other hand, our head attachment error was 44% for prepositions. Given the relatively large number of prepositions found in the test set, it is important to get the preposition attachment right to achieve a higher mark in this language. The obvious solution is to have a feature that connects the head of a preposition to the child of the preposition. However, such a feature effects the edge based factoring and the decoding algorithm, and we will be forced to modify the MST algorithm in some ways.

5.2 Swedish

Due to the memory constraint on the computer, we did not use the additional features for Swedish and our feature heavily relied on the CPOS tag. At the same time, we have noticed that relatively higher performance of our parser compared to the average coincides with the bigger tag set for CPOS for this corpus. This suggests that we should be using more fine grained POS in other languages.

5.3 Turkish

The difficulty with parsing Turkish stems from the large unlabeled attachment error rate on the nouns

Language	LAS	AV	SD
Arabic	62.83%	59.92%	6.53
Danish	75.81%	78.31%	5.45
Slovene	64.57%	65.61%	6.78
Spanish	73.17%	73.52%	8.41
Swedish	79.49%	76.44%	6.46
Turkish	54.23%	55.95%	7.71
Language	UAS	AV	SD
Arabic	74.27%	73.48%	4.94
Danish	81.72%	84.52%	4.29
Slovene	74.88%	76.53%	4.67
Spanish	77.58%	77.76%	7.81
Swedish	86.62%	84.21%	5.45
Turkish	68.77%	69.35%	5.51

Table 3: Labeled and Unlabeled Attachment Score

(39%). Since the nouns are the most frequently occurring words in the test set (2209 out of 5021 total), this seems to make Turkish the most challenging language for any system in the shared task. On the average, there are 1.8 or so verbs per sentence, and nouns have a difficult time attaching to the correct verb or postposition. This, we think, indicates that there are morphological features or word ordering features that we really need in order to disambiguate them.

6 Future Work

As well as making use of fine-grained POS tags and other morphological features, given the error analysis on Arabic, we would like to add features that are dependent on two or more edges.

6.1 Bottom-Up Non-Projective Parsing

In order to incorporate features which depend on other edges, we propose Bottom-Up Non-Projective Parsing. It is often the case that dependency relations can be ordered by how close one relation is to the root of dependency tree. For example, the dependency relation between a determiner and a noun should be decided before that between a preposition and a noun, and that of a verb and a preposition, and so on. We can use this information to do bottom-up parsing.

Suppose all words have a POS tag assigned to them, and every edge labeled with a dependency relation is attached to a specific POS tag at the end point. Also assume that there is an ordering of POS tags such that the edge going to the POS tag needs be decided before other edges. For example, (1) determiner, (2) noun, (3) preposition, (4) verb would be one such ordering. We propose the following algorithm:

- Assume we have tokens as nodes in a graph and no edges are present at first. For example, we have tokens ”I”, ”ate”, ”with”, ”a”, ”spoon”, and no edges between them.
- Take the POS tag that needs to be decided next. Find all edges that go to each token labeled with this POS tag, and put them in the graph. For example, if the POS is noun, put edges from ”ate” to ”I”, from ”ate” to ”spoon”, from ”with” to ”I”, from ”with” to ”spoon”, from ”I” to ”spoon”, and from ”spoon” to ”I”.
- Run the CLE algorithm on this graph. This selects the highest incoming edge to each token with the POS tag we are looking at, and remove cycles if any are present.
- Take the resulting forests and for each edge, bring the information on the child node to the parent node. For example, if this time POS was noun, and there is an edge to a preposition ”with” from a noun ”spoon”, then ”spoon” is absorbed by ”with”. Note that since no remaining dependency relation will attach to ”spoon”, we can safely ignore ”spoon” from now on.
- Go back and repeat until no POS is remaining and we have a dependency tree. Now in the next round, when deciding the score of the edge from ”ate” to ”with”, we can use the all information at the token ”with”, including ”spoon”.

7 Conclusion

We have extended non-projective unlabeled dependency parsing (McDonald et al., 2005) to a very simple non-projective labeled dependency and showed that the parser performs reasonably well with small number of features and just one iteration of training. Based on the analysis of the Arabic parsing results, we have proposed a bottom-up non-projective labeled dependency parsing algorithm that allows us to use features dependent on more than one edge, with very little disadvantage compared to the original algorithm.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*. Kluwer Academic Publishers, Dordrecht.

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. “Floresta sintá(c)tica”: a treebank for Portuguese. In *Proc. of the Third Intern. Conf. on Language Resources and Evaluation (LREC)*, pages 1698–1703.

N. B. Atalay, K. Oflazer, and B. Say. 2003. The annotation process in the Turkish treebank. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proc. of the First Workshop on Treebanks and Linguistic Theories (TLT)*.

R. Bunescu and R. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proc. of the Joint Conf. on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

Y.J. Chu and T.H. Liu. 1965. On the shortest arborescence of a directed graph. In *Science Sinica*, page 14:13961400.

M. Civit Torruella and M^{a} A. Martí Antonín. 2002. Design principles for a Spanish treebank. In *Proc. of the First Workshop on Treebanks and Linguistic Theories (TLT)*.

M. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proc. of the 42rd Annual Meeting of the ACL*.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. In *JMLR*.

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of the 43rd Annual Meeting of the ACL*.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proc. of the Fifth Intern. Conf. on Language Resources and Evaluation (LREC)*.

J. Edmonds. 1967. Optimum branchings. In *Journal of Research of the National Bureau of Standards*, page 71B:233240.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

Y. Kawata and J. Bartels. 2000. Stylebook for the Japanese treebank in VERBMOBIL. Verbmobil-Report 240, Seminar für Sprachwissenschaft, Universität Tübingen.

M. T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In *Proc. of the Second Workshop on Treebanks and Linguistic Theories (TLT)*.

G. Leonidas. 2003. Arborescence optimization problems solvable by edmonds algorithm. In *Theoretical Computer Science*, page 301:427 437.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of the Joint Conf. on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

J. Nilsson, J. Hall, and J. Nivre. 2005. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. In *Proc. of the NODALIDA Special Session on Treebanks*.

J. Nivre and J. Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of the 43rd Annual Meeting of the ACL*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15.

K. Simov and P. Osenova. 2003. Practical annotation scheme for an HPSG treebank of Bulgarian. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*, pages 17–24.

K. Simov, P. Osenova, A. Simov, and M. Kouylekov. 2005. Design and implementation of the Bulgarian HPSG-based treebank. In *Journal of Research on Language and Computation – Special Issue*, pages 495–522. Kluwer Academic Publishers.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

The Exploration of Deterministic and Efficient Dependency Parsing

Yu-Chieh Wu
Dept. of Computer Science and Information Engineering
National Central University
Taoyuan, Taiwan
bcbb@db.csie.ncu.edu.tw

Yue-Shi Lee
Dept. of Computer Science and Information Engineering
Ming Chuan University
Taoyuan, Taiwan
lees@mcu.edu.tw

Jie-Chi Yang
Graduate Institute of Network Learning Technology
National Central University
Taoyuan, Taiwan
yang@cl.ncu.edu.tw

Abstract

In this paper, we propose a three-step multilingual dependency parser, which generalizes an efficient parsing algorithm at first phase, a root parser and post-processor at the second and third stages. The main focus of our work is to provide an efficient parser that is practical to use with combining only lexical and part-of-speech features toward language independent parsing. The experimental results show that our method outperforms Maltparser in 13 languages. We expect that such an efficient model is applicable for most languages.

1 Introduction

The target of dependency parsing is to automatically recognize the head-modifier relationships between words in natural language sentences. Usually, a dependency parser can construct a similar grammar tree with the dependency graph. In this year, CoNLL-X shared task (Buchholz et al., 2006) focuses on multilingual dependency parsing without taking the language-specific knowledge into account. The ultimate goal of this task is to design an ideal multilingual portable dependency parsing system.

To accomplish the shared task, we present a very light-weight and efficient parsing model to the 13 distinct treebanks (Hajič et al., 2004; Simov et al., 2005; Simov and Osenova, 2003; Chen et al., 2003; Böhmová et al., 2003; Kromann 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit and Martí 2002; Nivre et al., 2005; Oflazer et al., 2003; Atalay et al., 2003) with a three-step process, Nivre's algorithm (Nivre, 2003), root parser, and post-processing. Our method is quite different from the conventional three-pass processing, which usually exhaustively processes the whole dataset three times, while our method favors examining the "un-parsed" tokens, which incrementally shrink. At the beginning, we slightly modify the original parsing algorithm (proposed by (Nivre, 2003)) to construct the initial dependency graph. A root parser is then used to recognize root words, which were not parsed during the previous step. At the third phase, the post-processor (which is another learner) recognizes the still un-parsed words. However, in this paper, we aim to build a multilingual portable parsing model without employing deep language-specific knowledge, such as lemmatization, morphologic analyzer etc. Instead, we only make use of surface lexical and part-of-speech (POS) information. Combining these shallow features, our parser achieves a satisfactory result for most languages, especially Japanese.

In the remainder of this paper, Section 2 describes the proposed parsing model, and Section 3 lists the experimental settings and results. Section 4 presents the discussion and analysis of our parser with three selected languages. In Section 5, we draw the future direction and conclusion.

2 System Description

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 241–245, New York City, June 2006.

Over the past decades, many state-of-the-art parsing algorithm were proposed, such as head-word lexicalized PCFG (Collins, 1998), Maximum Entropy (Charniak, 2000), Maximum/Minimum spanning tree (MST) (McDonald et al., 2005), Bottom-up deterministic parsing (Yamada and Matsumoto, 2003), and Constant-time deterministic parsing (Nivre, 2003). Among them, the Nivre's algorithm (Nivre, 2003) was shown to be most efficient method, which only costs at most 2n transition actions to parse a sentence ($O(n^3)$ for the bottom-up or MST approaches). Nivre's method is mainly consists of four transition actions, Left/Right/Reduce/Shift. We further extend these four actions by dividing the "reduce" into "reduce" and "sleep (reduce-but-shift)" two actions. Because the too early reduce action makes the following words difficult to find the parents. Thus, during training, if a word which is the child of the top of the stack, it is then assigned to the "sleep" category and pushed into stack, otherwise, the conventional reduce action is applied. Besides, we do not arrange these transition actions with priority order, instead, the decision is made by the classifier. The overall parsing model can be found in Figure 1. Table 1 lists the detail system spec of our model.

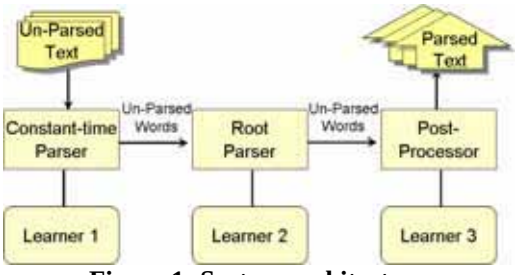


Figure 1: System architecture

Table 1: Overall parsing system summary

Ⅰ.	Parsing Algorithm:	1. Nivre's Algorithm (Nivre, 2003) 2. Root Parser 3. Exhaustive-based Post-processing
Ⅱ.	Parser Characteristics:	1. Top-down + Bottom-up 2. Deterministic + Exhaustive 3. Labeling integrated 4. Non-Projective
Ⅲ.	Learner:	SVMLight (Joachims, 1998) (1) One-versus-One (2) Linear Kernel
Ⅳ.	Feature Set:	1. Lexical (Unigram/Bigram) 2. Fine-grained POS and Coarse grained BiCPOS
Ⅴ.	Post-Processing:	Another learner is used to re-recognize heads in stacks
Ⅵ.	Additional/External Resources:	Non-Used

2.1 Constant-time Parser and Analysis

The Nivre's algorithm makes use of a stack and an input list to model the word dependency relations via identifying the transition action of the top token on the stack (*Top*) and the next token of the input list (*Next*). Typically a learning algorithm can be used to recognize these actions via encoding features of the two terms (*Top* and *Next*). The "Left" and "Reduce" pops the *Top* from stack whereas the "Right", "Reduce-But-Shift", and "Shift" push token *Next* into the top of stack. Nivre (Nivre, 2003) had proved that this algorithm can accomplish dependency parsing at most 2*n* transition actions.

Although, the Nivre's algorithm is much more efficient than the others, it produces three problems.

1. It does not explicitly indicate which words are the roots.
2. Some of the terms in the stack do not belong to the root but still should be parsed.
3. It always only compares the *Top* and *Next* words.

The problem (2) and (3) are complement with each other. A straightforward way resolution is to adopt the exhaustive parsing strategy (Covington, 2001). Unfortunately, such a brute-force way may cause exponential training and testing spaces, which is impractical to apply to the large-scale corpus, for example, the Czech Treebank (1.3 million words). To overcome this and keep the efficiency, we design a post-processor that re-cycles the residuum in the stack and re-identify the heads of them. Since most of the terms (90-95%) of the terms had be processed in previous stages, the post-processor just exhaustively parses a small part. In addition, for problem (1), we propose a root parser based on the parsed result of the Nivre's algorithm. We discuss the root-parser and post-processor in the next two subsections.

2.2 Root Parser

After the first stage, the stack may contain root and un-parsed words. The root parser identifies the root word in the stack. The main advantage of this strategy could avoid sequential classification process, which only focuses on terms in the stack.

We build a classifier, which learns to find root word based on encoding context and children features. However, most of the dependency relations were constructed at the first stage. Thus, we have more sufficient head-modifier information rather

than only taking the contexts into account. The used features are listed as follows.

Neighbor terms,bigrams,POS,BiCPOS (+/-2 window)
Left most child term, POS, Bigram, BiCPOS
Right most child term, POS, Bigram, BiCPOS

2.3 Post-Processing

Before post-processing, we remove the root words from stack, which were identified by root-parser. The remaining un-parsed words in stack were used to construct the actual dependency graph via exhaustive comparing with parsed-words. It is necessary to build a post-processor since there are about 10% un-parsed words in each training set. We provide the un-parsed rate of each language in Table 2 (the r.h.s. part).

By applying previous two steps (constant-time parser and root parser) to the training data, the remaining un-parsed tokens were recorded. Not only using the forward parsing direction, the backward direction is also taken into account in this statistics. Averagely, the un-parsed rates of the forward and backward directions are 13% and 4% respectively. The back ward parsing often achieves lower un-parsed rate among all languages (except for Japanese and Turkish).

To find the heads of the un-parsed words, we copy the whole sentence into the word list again, and re-compare the un-parsed tokens (in stack) and all of the words in the input list. Comparing with the same words is disallowed. The comparing process is going on until the actual head is found. Acquiescently, we use the nearest root words as its head. Although such a brute force way is time-consuming. However, it only parses a small part of un-parsed tokens (usually, 2 or 3 words per sentence).

2.4 Features and Learners

For the constant-time parser of the first stage, we employ the features as follows.

Basic features:

Top.word,*Top*.pos,*Top*.lchild.pos,*Top*.lchild.relation,
Top.rchild.pos, *Top*.rchild.relation,*Top*.head.pos,
Top.head.relation,
Next.word, *Next*.pos, *Next*.lchild.pos,
Next.lchild.relation, $Next_{+1}$.pos, $Next_{+2}$.pos, $Next_{+3}$.pos

Enhanced features:

Top.bigram,*Top*.bicpos,*Next*.bigram,*Next*.bicpos,
$Next_{+1}$.word,$Next_{+2}$.word,$Next_{+3}$.word

In this paper, we use the support vector machines (SVM) (Joachims, 1998) as the learner. SVM is widely used in many natural language processing (NLP) areas, for example, POS tagging (Wu et al., 2006). However, the SVM is a binary classifier which only recognizes true or false. For multiclass problem, we use the so-called one-versus-one (OVO) method with linear kernel to combine the results of each pairwise subclassifier. The final class in testing phase is mainly determined by majority voting.

For all languages, our parser uses the same settings and features. For all the languages (except Japanese and Turkish), we use backward parsing direction to keep the un-parsed token rate low.

3 Experimental Result

3.1 Dataset and Evaluation Metrics

The testing data is provided by the (Buchholz et al., 2006) which consists of 13 language treebanks. The experimental results are mainly evaluated by the unlabeled and labeled attachment scores. The CoNLL also provided a perl-scripter to automatic compute these rates.

3.2 System Results

Table 2 presents the overall parsing performance of the 13 languages. As shown in Table 2, we list two parsing results at the second and third columns (new and old). It is worth to note that the result B is produced by removing the enhanced features and the post-processing step from our parser, while the result A is the complete use of the enhanced features and the overall three-step parsing. In this year, we submit result B to the CoNLL shared task due to the time limitation.

In addition, we also apply the Maltparser, which is implemented with the Nivre's algorithm (Nivre, 2003) to be compared. The Maltpaser also includes the SVM and memory-based learner (MBL). Nevertheless, it does not optimize the SVM where the training and testing times are too long to be compared even the linear kernel is used. Therefore we use the default MBL and feature model 3 (M3) in this experiment. We also perform the significant test to evaluate the statistical difference among the three results. If the answer is "Yes", it means the two systems are significant difference under at least 95% confidence score ($p < 0.05$).

Table 2: A general statistical table of labeled attachment score, test and un-parsed rate (percentage)

	A (New result)	B (Old result)	C (Maltparser)	Statistic test			Un-Parsed Rate	
				A vs. B	B vs. C	A vs. C	Forward	Backward
Arabic	63.75	**63.81**	54.11	No	Yes	Yes	10.3	**1.4**
Chinese	**81.25**	74.81	73.92	Yes	No	Yes	4.01	**2.3**
Czech	**71.24**	59.36	59.36	Yes	No	Yes	16.1	**5.6**
Danish	**79.52**	78.38	77.31	No	No	No	12.8	**2.5**
Dutch	68.45	**68.45**	63.61	No	Yes	Yes	18.4	**9.8**
German	**79.57**	76.52	76.52	Yes	No	Yes	12.7	**9.2**
Japanese	**91.43**	90.11	89.07	Yes	No	Yes	**1.1**	4.4
Portugese	81.33	**81.47**	75.38	No	Yes	Yes	24.3	**3.17**
Slovene	**68.41**	67.83	55.04	No	Yes	Yes	14.9	**5.5**
Spanish	**74.65**	72.99	72.81	Yes	No	Yes	20	**0.5**
Swedish	**79.53**	71.72	76.28	Yes	Yes	Yes	19.1	**2.8**
Turkish	**55.33**	55.09	52.18	No	Yes	Yes	**2.5**	4
Bulgarian	**81.23**	79.73	79.73	No	No	No	15.7	**1.2**
AVG	75.05	72.32	69.64				13.22	4.02

4 Discussion

4.1 Analysis of Overview Aspect

Although our method is efficient for parsing that achieves satisfactory result, it is still away from the state-of-the-art performance. Many problems give rise to not only the language-specific characteristics, but also the parsing strategy. We found that our method is weak to the large-scale training size and large dependency class datasets, for example, German (Brants et al., 2002) and Czech. For Dutch, we observe that the large non-projective tokens and relations in this set. Overall, we conclude the four main limitations of our parsing model.

1. Unbalanced and large dependency relation classes
2. Too fine or coarse POS tag
3. Long sentences and non-projective token rates
4. Feature engineering and root accuracy

The main reason of the first problem is still caused by the unbalanced distribution of the training data. Usually, the right-action categories obtain much fewer training examples. For example, in the Turkish data, 50 % of the categories receive less than 0.1% of the training examples, 2/3 are the right dependency group. For the Czech, 74.6% of the categories receive less than 0.1% of the training examples.

Second, the too fine grained size of POS tag set often cause the features too specific that is difficult to be generalized by the learner. Although we found the grained size is not the critical factor of our parser, it is closely related to the fourth problem, feature engineering. For example, in Chinese (Chen et al., 2003), there are 303 fine grained POS types which achieves better result on the labeled attachment score is higher than the coarse grained (81.25 vs. 81.17). Intuitively, the feature combinations deeply affect the system performance (see A vs. C where we extend more features than the original Nivre's algorithm).

Problem 3 exposes the disadvantage of our method, which is weak to identify the long distance dependency. The main reason is resulted from the Nivre's algorithm in step 1. This method is quite sensitive and non error-recovered since it is a deterministic parsing strategy. Abnormal or wrong push or pop actions usually cause the error propagation to the remaining words in the list. For example, there are large parts of errors are caused by too early reduce or missed left arc makes some words could not find the actual heads. On the contrary, one can use an *N*-best selection to choose the optimal dependency graph or applying MST or exhaustive parsing schema. Usually, these approaches are quite inefficient which requires at least $O(n^3)$.

Finally, in this paper, we only take the surface lexical word and POS tag into account without employing the language-specific features, such as Lemma, Morph…etc. Actually, it is an open question to compile and investigate the feature engineering. On the other hand, we also find the performance of the root parser in some languages is poor. For example, for Dutch the root precision rate is only 38.52, while the recall rate is 76.07. It indicates most of the words in stack were wrongly recognized as root. This is because there are substantially un-parsed rate that left many un-parsed words remain in stack. One way to remedy the problem can adjust the root parser to independently identify root word by sequential word classification at first step and then apply the Nivre's algorithm. We left the comparison of the issue as future work.

4.2 Analysis of Specific View

We select three languages, Arabic, Japanese, and Turkish to be more detail analysis. Figure 2 illustrates the learning curve of the three languages and Table 3 summarizes the comparisons of "fine vs. coarse" POS types and "forward vs. backward" parsing directions.

For the three languages, we found that most of the errors frequently appear to the noun POS tags which often denominate half of the training set. In Turkish, the lower performance on the noun POS attachment rate deeply influents the overall parsing. For example, the error rate of Noun in Turkish is 39% which is the highest error rate. On the contrary, the head error rates fall in the middle rank for the other two languages.

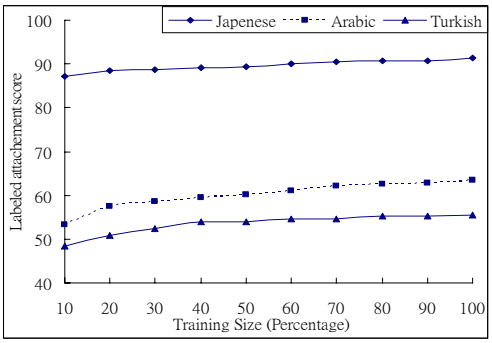


Figure 2: Learning curve of the three datasets

Table 3: Parsing performance of different grained POS tags and forward/backward parsing directions

	Parsing direction	LA-Score		POS grained	LA-Score
Ja	**Forward**	**91.35**	Forward	**Fine**	**91.35**
	Backward	85.75		Coarse	91.25
Ar	Forward	60.62	Backward	Fine	63.55
	Backward	**63.55**		**Coarse**	**63.63**
Tu	Forward	55.47	Forward	Fine	55.47
	Backward	**55.59**		**Coarse**	**55.59**

In Turkish, we also find an interesting result where the recall rate of the distance=2 parsing (56.87) is lower than distance=3-6, and >7 (62.65, 57.83). In other words, for Turkish, our parser failed to recognize the distance=2 dependency relations. For the other languages, usually the identification rate of the longer distance parsing should be lower than the smaller distance. Thus, a future work to parsing Turkish, should put more emphasis on improving not only the noun POS type, but also the distance=2 parsing.

Besides, the root parsing accuracy is also an important factor to most languages. In Japanese, although our parser achieves more than 97% left/right arc rates. However, for the root word precision rate is quite lower (85.97). Among all dependency relation classification rates, the root class usually locates in the lowest rank for the three languages.

5 Conclusion and Future Remarks

Dependency parsing is one of the most important issues in NLP community. This paper presents and analyzes the impact of the efficient parsing model that only combines with lexical and part-of-speech information. To go language-independent, we did not tune any parameter settings in our model and exclude most of the language-dependent feature set, which provided by the CoNLL (Buchholz et al., 2006). The main focus of our work coincides with the target goal of the CoNLL shared task, i.e., go multilingual dependency parsing without taking the language-specific knowledge into account. A future work on the deterministic parsing strategy is to convert the existing model toward *N*-best parsing.

References

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing, *In Proceedings of the Tenth Conf. on Computational Natural Language Learning CoNLL-X*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. *In NAACL*, pages 132-139.

Michael Collins. 1998. Head-driven statistical models for natural language processing. Ph.D. thesis. University of Pennsylvania.

Michael A. Covington. 2001. A fundamental Algorithm for Dependency Parsing. *In Proceedings of the Annual ACM Southeast Conference*, pages 95-102.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. *In COLING*, pages 340-345.

Thornsten Joachims. 1998. Text categorization with support vector machines: learning with many relevant features. *In ECML*, pages 137-142.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers, *In ACL*, pages 91-98.

Joakim Nivre. 2003. An Efficient Algorithm for Projective Dependency Parsing. *In Proceedings of the International Workshop on Parsing Technology*, pages 149-160.

Yu C. Wu, Chia H. Chang, and Yue S. Lee. 2006. A General and Multi-lingual Phrase Chunking Model based on Masking Method. *In CICLING*, pages 144-155.

Hiroyasu Yamada, and Yuji Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. *In Proceedings of the International Workshop on Parsing Technology*, pages 195-206.

Dependency Parsing as a Classification Problem

Deniz Yuret
Koç University
Istanbul, Turkey
dyuret@ku.edu.tr

Abstract

This paper presents an approach to dependency parsing which can utilize any standard machine learning (classification) algorithm. A decision list learner was used in this work. The training data provided in the form of a treebank is converted to a format in which each instance represents information about one word pair, and the classification indicates the existence, direction, and type of the link between the words of the pair. Several distinct models are built to identify the links between word pairs at different distances. These models are applied sequentially to give the dependency parse of a sentence, favoring shorter links. An analysis of the errors, attribute selection, and comparison of different languages is presented.

1 Introduction

This paper presents an approach to supervised learning of dependency relations in a language using standard machine learning techniques. The treebanks (Hajič et al., 2004; Chen et al., 2003; Böhmová et al., 2003; Kromann, 2003; van der Beek et al., 2002; Brants et al., 2002; Kawata and Bartels, 2000; Afonso et al., 2002; Džeroski et al., 2006; Civit Torruella and Martí Antonín, 2002; Nilsson et al., 2005; Oflazer et al., 2003; Atalay et al., 2003) provided for the CoNLL shared task(Buchholz et al., 2006) were converted to a set of instances each of which consists of the attributes of a candidate word pair with a classification that indicates the existence, direction and type of the dependency link between the pair.

An initial model is built to identify dependency relations between adjacent word pairs using a decision list learning algorithm. To identify longer distance relations, the adjacent modifiers are dropped from the sentence and a second order model is built based on the word pairs that come into contact. A total of three models were built using this technique successively and used for parsing.

All given attributes are considered as candidates in an attribute selection process before building each model. In addition, attributes indicating suffixes of various lengths and character type information were constructed and used.

To parse a given sentence, the models are applied sequentially, each one considering candidate word pairs and adding new links without deleting the existing links or creating conflicts (cycles or crossings) with them. Thus, the algorithm can be considered a bottom-up, multi-pass, deterministic parser. Given a candidate word pair, the models may output "no link", or give a link with a specified direction and type. Thus labeling is an integrated step. Word pair candidates that may form cycles or crossings are never considered, so the parser will only generate projective structures.

Section 2 gives the details of the learning algorithm. Section 3 describes the first pass model of links between adjacent words. Section 4 details the approach for identifying long distance links and presents the parsing results.

Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X),
pages 246–250, New York City, June 2006.

2 The Learning Algorithm

The Greedy Prepend Algorithm (Yuret and Ture, 2006) was used to build decision lists to identify dependency relations. A decision list is an ordered list of rules where each rule consists of a pattern and a classification (Rivest, 1987). The first rule whose pattern matches a given instance is used for its classification. In our application the pattern specifies the attributes of the two words to be linked such as parts of speech and morphological features. The classification indicates the existence and the type of the dependency link between the two words.

Table 1 gives a subset of the decision list that identifies links between adjacent words in German. The class column indicates the type of the link, the pattern contains attributes of the two candidate words X and Y, as well as their neighbors (XL1 indicates the left neighbor of X). For example, given the part of speech sequence APPR-ART-NN, there would be an NK link between APPR and ART (matches rule 3), but there would be no link between ART and NN (rule 1 overrides rule 2).

Rule	Class	Pattern
1	NONE	XL1:postag=APPR
2	L:NK	X:postag=ART Y:postag=NN
3	R:NK	X:postag=APPR
4	NONE	

Table 1: A four rule decision list for adjacent word dependencies in German

The average training instance for the dependency problem has over 40 attributes describing the two candidate words including suffixes of different lengths, parts of speech and information on neighboring words. Most of this information may be redundant or irrelevant to the problem at hand. The number of distinct attribute values is on the order of the number of distinct word-forms in the training set. GPA was picked for this problem because it has proven to be fairly efficient and robust in the presence of irrelevant or redundant attributes in previous work such as morphological disambiguation in Turkish (Yuret and Ture, 2006) and protein secondary structure prediction (Kurt, 2005).

3 Dependency of Adjacent Words

We start by looking at adjacent words and try to predict whether they are linked, and if they are, what type of link they have. This is a nice subproblem to study because: (i) It is easily converted to a standard machine learning problem, thus amenable to common machine learning techniques and analysis, (ii) It demonstrates the differences between languages and the impact of various attributes. The machine learning algorithm used was GPA (See Section 2) which builds decision lists.

Table 2 shows the percentage of adjacent tokens that are linked in the training sets for the languages studied[1]. Most languages have approximately half of the adjacent words linked. German, with 42.15% is at the low end whereas Arabic and Turkish with above 60% are at the high end. The differences may be due to linguistic factors such as the ubiquity of function words which prefer short distance links, or it may be an accident of data representation: for example each token in the Turkish data represents an inflectional group, not a whole word.

Arabic	61.02	Japanese	54.81
Chinese	56.59	Portuguese	50.81
Czech	48.73	Slovene	45.62
Danish	55.93	Spanish	51.28
Dutch	55.54	Swedish	48.26
German	42.15	Turkish	62.60

Table 2: Percentage of adjacent tokens linked.

3.1 Attributes

The five attributes provided for each word in the treebanks were the wordform, the lemma, the coarse-grained and fine-grained parts of speech, and a list of syntactic and/or morphological features. In addition I generated two more attributes for each word: suffixes of up to n characters (indicated by suffix[n]), and character type information, i.e. whether the word contains any punctuation characters, upper case letters, digits, etc.

Two questions to be answered empirically are: (i) How much context to include in the description of each instance, and (ii) Which attributes to use for each language.

[1] Including non-scoring tokens

Table 3 shows the impact of using varying amounts of context in Spanish. I used approximately 10,000 instances for training and 10,000 instances for testing. Only the postag feature is used for each word in this experiment. As an example, consider the word sequence $w_1 \ldots w_i w_{i+1} \ldots w_n$, and the two words to be linked are w_i and w_{i+1}. Context=0 means only information about w_i and w_{i+1} is included, context=1 means we also include w_{i-1} and w_{i+2}, etc. The table also includes the number of rules in each decision list. The results are typical of the experiments performed with other languages and other attribute combinations: there is a statistically significant improvement going from context=0 to context=1. Increasing the context size further does not have a significant effect.

Context	Rules	Accuracy
0	161	83.17
1	254	87.31
2	264	87.05
3	137	87.14

Table 3: Context size vs. accuracy in Spanish.

A number of experiments were run to determine the best attribute combinations for each language. Table 4 gives a set of results for single attributes in Spanish. These results are based on 10,000 training instances and all experiments use context=1. Postag was naturally the most informative single attribute on all languages tested, however the second best or the best combination varied between languages. Suffix[3] indicates all suffixes up to three characters in length. The FEATS column was split into its constituent features each of which was treated as a binary attribute.

Attributes	Rules	Accuracy
postag	254	87.31
cpostag	154	85.72
suffix[3]	328	77.15
lemma	394	76.78
form	621	75.06
feats	66	71.95
ctype	47	53.40

Table 4: Attributes vs. accuracy in Spanish.

There are various reasons for performing attribute selection. Intuitively, including more information should be good, so why not use all the attributes? First, not every machine learning algorithm is equally tolerant of redundant or irrelevant attributes. Naive Bayes gets 81.54% and C4.5 gets 86.32% on the Spanish data with the single postag attribute using context=1. One reason I chose GPA was its relative tolerance to redundant or irrelevant attributes. However, no matter how robust the algorithm, the lack of sufficient training data will pose a problem: it becomes difficult to distinguish informative attributes from non-informative ones if the data is sparse. About half of the languages in this study had less than 100,000 words of training data. Finally, studying the contribution of each attribute type in each language is an interesting research topic in its own right. The next section will present the best attribute combinations and the resulting accuracy for each language.

3.2 Results

Language	Attributes	Accuracy
Arabic	ALL	76.87
Chinese	postag, cpostag	84.51
Czech	postag, lemma	79.25
Danish	postag, form	86.96
Dutch	postag, feats	85.36
German	postag, form	87.97
Japanese	postag, suffix[2]	95.56
Portuguese	postag, lemma	90.18
Slovene	ALL	85.19
Spanish	postag, lemma	89.01
Swedish	postag, form	83.20
Turkish	ALL	85.27

Table 5: Adjacent word link accuracy.

Table 5 gives the best attribute combinations for determining adjacent word links for each language studied. The attribute combinations and the corresponding models were determined using the training sets, and the accuracy reported is on the test sets. These attribute combinations were used as part of the model in the final evaluation. I used context=1 for all the models. Because of time limitations attribute combinations with more than two attributes

could not be tested and only the first 100,000 training instances were used. Exceptions are indicated with "ALL", where all attributes were used in the model – these are cases where using all the attributes outperformed other subsets tried.

For most languages, the adjacent word link accuracy is in the 85-90% range. The outliers are Arabic and Czech at the lower end, and Japanese at the higher end. It is difficult to pinpoint the exact reasons: Japanese has the smallest set of link types, and Arabic has the greatest percentage of adjacent word links. Some of the differences between the languages come from linguistic origins, but many are due to the idiosyncrasies of our particular data set: number of parts of speech, types of links, quality of the treebank, amount of data are all arbitrary factors that effect the results. One observation is that the ranking of the languages in Table 5 according to performance is close to the ranking of the best results in the CoNLL shared task – the task of linking adjacent words via machine learning seems to be a good indicator of the difficulty of the full parsing problem.

4 Long Distance Dependencies

Roughly half of the dependency links are between non-adjacent words in a sentence. To illustrate how we can extend the previous section's approach to long distance links, consider the phrase "kick the red ball". The adjacent word linker can only find the *red-ball* link even if it is 100% accurate. However once that link has been correctly identified, we can drop the modifier "red" and do a second pass with the words "kick the ball". This will identify the link *the-ball*, and dropping the modifier again leaves us with "kick ball". Thus, doing three passes over this word sequence will bring all linked words into contact and allow us to use our adjacent word linker. Table 6 gives the percentage of the links discovered in each pass by a perfect model in Spanish.

Pass:	1	2	3	4	5
Link%:	51.09	23.56	10.45	5.99	3.65

Table 6: Spanish links discovered in multiple passes.

We need to elaborate a bit on the operation of "dropping the modifiers" that lead from one pass to the next. After the discovery of the *red-ball* link in the above example, it is true that "red" can no longer link with any other words to the right (it cannot cross its own head), but it can certainly link with the words to the left. To be safe, in the next pass we should consider both *the-red* and *the-ball* as candidate links. In the actual implementation, given a partial linkage, all "potentially adjacent" word pairs that do not create cycles or link crossings were considered as candidate pairs for the next pass.

There are significant differences between the first pass and the second pass. Some word pairs will rarely be seen in contact during the first pass (e.g. "kick ball"). Maybe more importantly, we will have additional "syntactic" context during the second pass, i.e. information about the modifiers discovered in the first pass. All this argues for building a separate model for the second pass, and maybe for further passes as well.

In the actual implementation, models for three passes were built for each language. To create the training data for the n'th pass, all the links that can be discovered with (n-1) passes are taken as given, and all word pairs that are "potentially adjacent" given this partial linkage are used as training instances. To describe each training instance, I used the attributes of the two candidate words, their surface neighbors (i.e. the words they are adjacent to in the actual sentence), and their syntactic neighbors (i.e. the words they have linked with so far).

To parse a sentence the three passes were run sequentially, with the whole sequence repeated twice[2]. Each pass adds new links to the existing partial linkage, but does not remove any existing links. Table 7 gives the labeled and unlabeled attachment score for the test set of each language using this scheme.

5 Conclusion

I used standard machine learning techniques to investigate the lower bound accuracy and the impact of various attributes on the subproblem of identifying dependency links between adjacent words. The technique was then extended to identify long distance dependencies and used as a parser. The model gives average results for Turkish and Japanese but

[2]This counterintuitive procedure was used because it gave the best results on the training set.

Language	LAS	UAS
Arabic	52.42	68.82
Chinese	72.72	78.37
Czech	51.86	66.36
Danish	71.56	78.16
Dutch	62.75	66.17
German	63.82	67.71
Japanese	84.35	87.31
Portuguese	70.35	79.46
Slovene	55.06	70.60
Spanish	69.63	73.89
Swedish	65.23	73.25
Turkish	60.31	71.54

Table 7: Labeled and unlabeled attachment scores.

generally performs below average. The lack of a specialized parsing algorithm taking into account sentence wide constraints and the lack of a probabilistic component in the model are probably to blame. Nevertheless, the particular decomposition of the problem and the simplicity of the resulting models provide some insight into the difficulties associated with individual languages.

References

A. Abeillé, editor. 2003. *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*. Kluwer Academic Publishers, Dordrecht.

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. "Floresta sintá(c)tica": a treebank for Portuguese. In *Proc. of the Third Intern. Conf. on Language Resources and Evaluation (LREC)*, pages 1698–1703.

N. B. Atalay, K. Oflazer, and B. Say. 2003. The annotation process in the Turkish treebank. In *Proc. of the 4th Intern. Workshop on Linguistically Interpreteted Corpora (LINC)*.

A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. 2003. The PDT: a 3-level annotation scenario. In Abeillé (Abeillé, 2003), chapter 7.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proc. of the First Workshop on Treebanks and Linguistic Theories (TLT)*.

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of the Tenth Conf. on Computational Natural Language Learning (CoNLL-X)*. SIGNLL.

K. Chen, C. Luo, M. Chang, F. Chen, C. Chen, C. Huang, and Z. Gao. 2003. Sinica treebank: Design criteria, representational issues and implementation. In Abeillé (Abeillé, 2003), chapter 13, pages 231–248.

M. Civit Torruella and M^{a} A. Martí Antonín. 2002. Design principles for a Spanish treebank. In *Proc. of the First Workshop on Treebanks and Linguistic Theories (TLT)*.

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene dependency treebank. In *Proc. of the Fifth Intern. Conf. on Language Resources and Evaluation (LREC)*.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic dependency treebank: Development in data and tools. In *Proc. of the NEMLAR Intern. Conf. on Arabic Language Resources and Tools*, pages 110–117.

Y. Kawata and J. Bartels. 2000. Stylebook for the Japanese treebank in VERBMOBIL. Verbmobil-Report 240, Seminar für Sprachwissenschaft, Universität Tübingen.

M. T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In *Proc. of the Second Workshop on Treebanks and Linguistic Theories (TLT)*.

Volkan Kurt. 2005. Protein structure prediction using decision lists. Master's thesis, Koç University.

J. Nilsson, J. Hall, and J. Nivre. 2005. MAMBA meets TIGER: Reconstructing a Swedish treebank from antiquity. In *Proc. of the NODALIDA Special Session on Treebanks*.

K. Oflazer, B. Say, D. Zeynep Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In Abeillé (Abeillé, 2003), chapter 15.

Ronald L. Rivest. 1987. Learning decision lists. *Machine Learning*, 2:229–246.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.

Deniz Yuret and Ferhan Ture. 2006. Learning morphological disambiguation rules for Turkish. In *HLT-NAACL 06*.

Author Index