Institute of Information and Communication Technologies
Bulgarian Academy of Sciences

Proceedings

of

The Twelfth Workshop on Treebanks and Linguistic Theories (TLT12)

Editors:
Sandra Kübler
Petya Osenova
Martin Volk

Supported by:

Ministry of Education and Science

Ontotext AD

13-14 December 2013
Sofia, Bulgaria

The workshop is supported by:

Ministry of Education and Science

Ontotext AD

ISBN: 978-954-91700-6-1

Preface

The Twelfth International Workshop on Treebanks and Linguistic Theories (TLT12) is held in Sofia, Bulgaria. Thus, after eleven years, it is returning to Bulgaria, where the first installment took place in 2002. Since this first workshop on the Black Sea coast, the TLT series has come a long way, and has become a venue for research on technical and linguistic issues involved in treebanking, as well as the usage of treebanks in Linguistics and Computational Linguistics. The workshop series has seen influential papers, such as the description of the TIGER Treebank (2002) and the Ancient Greek Treebank (2009). However, over the last decade, the workshop also became broader in its scope, including other levels of annotation such as frame semantics, co-reference, or events, to name only a few.

The program of TLT12 showcases this development. Both invited talks take a broad perspective into the issues of language resources. Antonio Branco focuses on meta-reliability factors in NLP, while Stefanie Dipper addresses the annotation specificities of historic corpora. The various topics at the workshop include: handling difficult linguistic phenomena, such as coordination classification in English (Wolfgang Maier and Sandra Kübler) or auxiliary fronting in German (Erhard Hinrichs and Kathrin Beck); minimizing human efforts in tree disambiguation of Polish (Katarzyna Krasnowska and Witold Kieraś), in multiword expression disambiguation tasks for English (Valia Kordoni), in treebank expansion for Hindi (Naman Jain et al.) and for Persian (Masood Ghayoomi and Jonas Kuhn); looking at annotator agreement for Norwegian (Helge Dyvik et. al); combining different parsers for achieving better syntactic processing for Bulgarian (Kiril Simov et al.) or for more efficiently detecting annotation-related errors with results for Urdu and Hindi (Narendra Annamaneni et. al); tools and methodologies for exploring treebanks, such as the TÜNDRA web tool, used for queries over the German TüBa-D/Z Treebank (Scott Martens), the methodology for reformatting the treebanks, tested on the Penn Treebank (Archna Bhatia et al.) or the approaches to the creation of reliable user-oriented "corpus research databases" (Erwin R. Komen)

This year's TLT features two invited talks, 12 regular papers, and a round table discussion on the future of treebanks. The round table is intended as a forum for researchers in the field to discuss and establish future directions for research on the intersection of treebanks, linguistic theories, and applications. It will help the workshop organizers of future TLTs to target a wider audience, and our hope is that it will also foster new, interdisciplinary research collaborations.

We would like to cordially thank all people: who are involved in organizing the workshop, who submitted papers, who did the reviewing, and who are participating in the round table. We hope you enjoy the workshop and the proceedings!

Sandra Kübler, Petya Osenova, Martin Volk

Workshop Organization

Invited Speakers

Stefanie Dipper, University of Bochum, Germany
Antonio Branco, University of Lisbon, Portugal

Program Chairs

Sandra Kübler, Indiana University, USA
Petya Osenova, Sofia University, Bulgaria
Martin Volk, University of Zurich, Switzerland

Program Committee

Yvonne Adesam, Gothenburg University, Sweden
Eckhard Bick, University of Southern Denmark, Denmark
Johan Bos, University of Amsterdam, The Netherlands
António Branco, University of Lisbon, Portugal
Koenraad De Smedt, Bergen University, Norway
Markus Dickinson, Indiana University, USA
Stefanie Dipper, Bochum University, Germany
Dan Flickinger, Stanford University, USA
Georgi Georgiev, Ontotext, Bulgaria
Anne Göhring, University of Zurich, Switzerland
Eva Hajičová, Charles University, Czech Republic
Iris Hendrickx, University of Nijmegen, The Netherlands
Erhard Hinrichs, University of Tuebingen, Germany
Valia Kordoni, Humboldt University, Germany
Amalia Mendes, University of Lisbon, Portugal
Detmar Meurers, University of Tuebingen, Germany
Yusuke Miyao, University of Tokyo, Japan
Kaili Muurisep, Tartu University, Estonia
Kemal Oflazer, Carnegie Mellon University, Qatar
Sebastian Padó, Heidelberg University, Germany
Marco Passarotti, Catholic University of the Sacred Heart, Italy
Kiril Simov, IICT-BAS, Bulgaria
Adam Przepiórkowski,Polish Academy of Sciences, Poland
Victoria Rosén, Bergen University, Norway
Caroline Sporleder, Saarland University, Germany
Manfred Stede, University of Potsdam, Germany
Gertjan van Noord, University of Groningen, The Netherlands
Heike Zinsmeister, Stuttgart University, Germany

Local Committee

Petya Osenova, Sofia University
Kiril Simov, IICT-BAS
Stanislava Kancheva, Sofia University
Georgi Georgiev, Ontotext
Borislav Popov, Ontotext

Table of Contents

Ensembling Dependency Parsers for Treebank Error Detection

Narendra Annamaneni, Riyaz Ahmad Bhat and Dipti Misra Sharma
Language Technology Research Centre, IIIT-Hyderabad, India
{narendra.annamaneni, riyaz.bhat}@research.iiit.ac.in, dipti@iiit.ac.in

Abstract

This paper describes a statistical approach to detect annotation errors in dependency treebanks. The approach is based on the ensembling of state-of-the-art dependency parsers. We see the motivation from the fact that if a parse, favoured by the parsers, contradicts human annotation, the contradiction either questions the consistency of the corpora on which the parsers were trained or the given human annotation is an error. We also prioritize the detected errors based on the confidence score values. The reported results (F-score) of our approach on the Urdu and Hindi treebanks are 41.20% and 69.37% respectively.

1 Introduction

The need for annotated corpora is widely acknowledged in the field of computational linguistics. Due to their importance for basic as well as advanced NLP applications, the last decade has seen a nearly exponential increase in the creation of these linguistic resources in a wide range of languages. These corpora are mainly manually annotated. The annotation process, however, can also be semi-automated. In either way, in the process of annotation, errors creep in due to various reasons which make these corpora inconsistent and less suitable for use. It has been noted that, the inconsistency and the errors in the annotated corpora limit their usage for reliable natural language processing [25]. Annotated corpora are desired to be consistent and error free for an optimal use. To overcome the problem of inconsistency, these resources are usually manually validated. However, manual validation is an expensive task both in terms of time and cost. In this context, we need tools which can assist experts by automatically detecting anomalies (potential errors).

Lately there have been some efforts towards building linguistic resources for Indian Languages. The main focus has been on building syntactic treebanks for Hindi and Urdu. This paper addresses the issue of consistency in these treebanks. We propose an approach for automatically detecting dependency errors in two of these treebank namely Hindi [11] and Urdu [9]. We use state-of-the-art dependency

parsers to detect errors in the aforementioned treebanks. The ensembling of these parsers is used to detect errors assuming that they will only agree on a parse if they had learned from a more or less consistent corpora. We also prioritized the detected errors based on the confidence score so that experts can actually validate those errors based on their priority. Even though we have reported on these two languages only, our approach can be easily adopted to any language.

The rest of the paper is organized as follows. Related work is described in section 2.In section 3, we describe our approach to detect errors using MaltParser[1], MstParser[2] and TurboParser[3]. Section 4 describes treebanks and data used in our experiments. Section 5 gives the details of experiments conducted and results obtained. Section 6 adds the discussion and future work. Section 7 concludes the paper.

2 Related Work

There has been an active research in this direction over last decade. Validation tools were built which detect inconsistencies in annotated corpora automatically. One such approach was proposed by [13] which uses variation n-grams to detect inconsistencies in constituency-based treebanks. Later it was extended to discontinuous constituency annotations in [14]. There have been some earlier efforts to detect anomalies in syntactic annotation mainly POS and Chunk by [15] and [25]. [26] proposed approach to produce gold standard parsed data automatically. Other note worthy approaches in the field of error detection are methods proposed in these works [12, 18].

In the context of Indian languages earlier effort on error detection was made by [5] which uses statistical module and rule based post processing module combinedly to detect anomalies. The statistical module used in the system is FBSM (Frequency Based Statistical Module) which uses frequencies to detect errors in a treebank. Later [3] proposed a different statistical module PBSM (Probability Based Statistical Module) to overcome data sparsity limitation existed in FBSM. Both the aforementioned methods use the rule based post processing module which uses robust rules derived from annotation guidelines and CPG framework. The role of this rule based system is to increase the precision. [1] extended the previous work further by replacing PBSM with EPBSM (Extended Probability Based Statistical Module). The statistical module in the aforementioned methods is not successful to learn the existing consistencies in a treebank. Recently [2] used a Maltparser to detect inconsistencies in treebank. In our approach we try to ensemble the available state-of-the-art dependency parsers to exploit the consistency in an annotated corpora. Our experiments are conducted on the Hindi and Urdu treebanks. The proposed system is able to detect errors across the entire treebanks.

[1] http://www.maltparser.org

[2] http://maltparser.org

[3] http://www.ark.cs.cmu.edu/TurboParser/

3 Our Approach

In order to present our approach, we will first discuss what differentiates error from inconsistency in an annotated corpora. The notion of consistency is related to the frequency of a linguistic object annotated i.e., if a markable is frequently annotated with similar information, the annotation can said to be consistent, while error is a notion relative to the annotation guidelines or the linguistic theory. Both the notions have, however, a strong dependence on each other. In principle, an error free corpora will also be consistent. However, a consistent annotation in a corpora can either be an error (deviation from guidelines) or a valid annotation. An inconsistent annotation, on the other hand, is less probable to be a valid annotation (exception could be some infrequent phenomena). In this work, we will address the issue of consistency in dependency treebanks. The problem of inconsistency could be due to various reasons like subjectivity, fatigue of annotators and instability of the annotation scheme. Issues like subjectivity and instability of guidelines depend on the maturity of the annotation project. Over time annotators mature in their decisions and the guidelines become stable. On the other hand, the issue of fatigue or heedlessness will always pose a problem. However, fortunately the issue of fatigue is observed to be an infrequent phenomena. The errors that creep in because of it would be highly skewed. Our assumption is that such issues have less bearing on the statistical predictions that a parser makes. We assume that a parser will learn the consistent annotations in a learner corpora and prune out the skewed inconsistencies. To this end, we have used three state-of-the-art statistical parsers namely Malt, MST and Turbo which use different learning and parsing strategies. We pool the predictions of these parsers to locate the errors in a treebank. In the next section, we will discuss our approach in detail.

3.1 Error Classification

For the purpose of our task, we use the three dependency parsers based on some compelling references which made us choose these parsers over the other existing parsers in the field. MaltParser [22] has been reported to be the best performing parser for Indian languages [8]. MST parser is reported as the second best parser as far as its performance of parsing Indian languages is concerned [4, 8]. The third parser that we choose to work with is Turbo parser [20]. [19] has reported it to give the state of the art results in Hindi parsing. Thus, our choice of these parsers is based on their performance of parsing Indian languages. In Section 5, we discuss the settings of these parsers in detail. We choose to test our approach on two almost stable Indian language treebanks namely the Hindi treebank and the Urdu treebank that are being developed parallelly. In Section 4, we will give a detailed overview of these treebanks. In order to detect errors across these treebanks, we use N-fold cross validation strategy to parse the entire treebanks. We have restricted N to 4 in this work. Using our strategy both the treebanks are parsed with the three parsers. Confidence score of a label is also computed in case of the Malt and MST parser

[21, 17]. After getting the parsed outputs of all the three parsers, we identify the errors based on the error detection strategy discussed in the following.

- **Type 1:** Type 1 errors are label errors defined based on the mismatch between the parsers and the human annotation. If the dependency label annotated by a human differs with the parsers, given that the parsers don't differ among themselves, the dependency label is treated as a possible error. However, there is no disagreement on the attachment among the parsers and the human annotation.

- **Type 2:** Type 2 errors are also label errors based on the confusion among all the parsers and human annotation. If the attachments marked by all the parsers and the human annotation agree on the same attachment and the dependency labels marked by all the parsers differ among themselves and differ with the human annotation,the dependency label is treated as a possible error. Even if any of the two parsers agree on the same label and differ with the label marked by the third parser and with the human annotation, the dependency label is treated as a possible error. The point here is that the disagreement among all the parsers and human annotation gives a clue that there is some confusion in a given instance, thus there is a possibility that human annotation can be an error. The confusion here can be due to rare occurrence of the given instance or lesser context.

- **Type 3:** Type 3 error cases are arc errors defined based on the mismatch between all the parsers and human annotation. If the attachment annotated by the human differs with the attachment which all the parsers are agreeing on, that attachment is treated as a possible error. The point here is that since all the parsers agree on the same attachment based on consistency in the treebank, it may be possible that the human annotated attachment is either an exception or an error. However, there is no condition on the labels marked by all the parsers and human annotation. We can't conclude anything if we compare the labels without the attachment agreement.

- **Type 4:** Type 4 errors are arc errors defined on the basis of the confusion among all the parsers and human annotation. If the attachments marked by all the parsers differ among themselves and all of them differ with the human annotated attachment, the attachment is treated as a possible error. Even if any of the two parsers agree on the same attachment and differ with the attachment marked by the third parser and with the human annotated, the attachment is treated as a possible error.

3.2 Prioritising the Errors

After detecting possible error cases in a treebank, the errors are prioritized based on the corresponding confidence scores calculated using MST parser and Malt parser.

The main aim of using the confidence scores is to further strengthen the possibility of a markable as an error and thus to prioritize them for validation. If the confidence scores produced by Malt and MST parsers, corresponding to *Type 1* and *Type 3* error cases are higher than a threshold[4], the probability of these cases being errorneous increases twofold. Not only are the parsers agreeing upon their parsed output but they are also confident about it. In case of other error types like *Type 2* and *Type 4*, the lack of agreement among the parsers over a parse suggests some inherent confusion in it, the low confidence scores below the threshold[5] further support the confusion in that particular parse, thus the possibility of such a parse being errorneous is increased. In this way, we prioritize the validation of detected errors in the treebank.

4 Treebanks

In this section, we give an overview of Indian Language treebanking. Currently dependency treebanks for four ILs, namely Hindi, Urdu, Bangla and Telegu, are under development. These treebanks are currently being developed following the annotation scheme based on the Computational Paninian Grammar (CPG) [6]. According to CPG, dependency relations, are marked between chunks. A chunk is a minimal, non-recursive structure consisting of a group of closely related words. Thus, in these treebanks a node in a dependency tree is represented by a chunk instead of a word. In these treebanks, dependency relations are mainly verb-centric. The relation that holds between a verb and its arguments is called a '*karaka*' relation. Besides *karaka* relations, dependency relations also exist between nouns (genitives), between nouns and their modifiers (adjectival modification, relativization), between verbs and their modifiers (adverbial modification including subordination). CPG provides an essentially syntactico-semantic dependency annotation, incorporating *karaka* (e.g., agent, theme, etc.), *non-karaka* (e.g. possession, purpose) and other (part of) relations. A complete tag-set of dependency relations based on CPG can be found in [7]. The ones starting with '*k*' are largely Paninian *karaka* relations, and are assigned to the arguments of a verb.

In this work, we address the issue of consistency in Hindi and Urdu treebanks. The Urdu treebank used here is under development. The Hindi treebank data we used here is the part of the larger Hindi treebank. We used the same Hindi treebank data used by [2]. The sizes of Hindi and Urdu treebanks used in our experiments are 67k and 160k respectively. The statistics of the treebanks used in our experiments are given in Table 1.

[4]Thresholds for Malt and Mst are 59% and 63%

[5]The lower bounds of thresholds for Malt and MST are 18% and 30%

Language	Sentences	Words / Sentences	Chunks /Sentences
Hindi	2869	12.82	7.95
Urdu	5230	13.17	8.17

Table 1: Table 1: Treebank Statistics

5 Experiments and Results

As discussed earlier, we are using three state-of-the-art dependency parsers in a stacked settings, for the task of error detection in Hindi and Urdu Treebanks. In this section, we first discuss the settings of these parsers and then give a detailed account of the results achieved using parser ensembling. We carried out our experiments with Malt, MST and Turbo parsers with the best possible settings proposed in the literature, on Hindi and Urdu parsing. In case of Malt parser, we used "nivreeager" algorithm and "LIBSVM" learner and same features settings proposed in [16] and [24]. Since both the Hindi and Urdu treebanks contain a considerable number of non-projective structures [10], we used the pseudo-projective algorithm as proposed by [23]. For Turbo parser, we used the settings proposed by [19] and we run the parser with basic mode for both Hindi and Urdu. In case of MST parser,settings that performed are second order non-projective with beam width (k-best parses) of 5 and default iterations of 10 which are taken from [4]. Apart from the basic parser settings, we also added a module in MST and Malt parsers to get the confidence scores for the dependency labels based on the methods proposed in [21, 17]. The parsing accuracies are reported in Table 2. Our results are in conformity with the earlier results on Hindi and Urdu found in the literature.

Language	Parsers	LAS (%)	UAS (%)	LAcc (%)
HINDI	Malt	78.23	90.74	80.77
	MST	65.68	86.68	68.02
	Turbo	77.23	89.88	79.78
URDU	Malt	74.45	87.45	77.92
	MST	64.35	85.59	67.65
	Turbo	74.05	86.65	77.71

Table 2: Table 2: Parsing Accuracies.

In Section 3, we defined the basis of our approach for error detection using parser ensembling strategy. We also defined a detailed schema for identification of different error types. Next, we discuss the identified errors using our approach and evaluate their correctness. To evaluate our approach, we extracted test samples from both, the Hindi and the Urdu treebank. 323 sentences and 100 sentences of varied lengths were extracted from the Hindi and Urdu treebanks respectively, and were validated by the expert annotators. On these sentences, errors identified by our approach evaluated against the corresponding validated annotations. The results are listed in Table 3 on both the Hindi and Urdu test sets.

Once the possible errors identified, we prioritized them based on the confusion scores given by Malt and MST parsers. If an identified error has a confusion score higher than a threshold, we can be sure about it being an error. This helps us to prioritize the most likely errors for the validation. This is important in a case where the whole treebank needs validation and the detected errors are too many for a trivial validation.

Category	**Hindi**	**Urdu**
Type 1	206	162
Type 2	123	168
Type 3	150	121
Type 4	33	20
System Output	512	472
Identified	340	130
Gold Errors	468	159
Precision	**66.40**	**27.54**
Recall	**72.64**	**81.76**
F-score	**69.37**	**41.20**

Table 3: Table 3: Results on Urdu/Hindi Treebanks

After the identification of inconsistencies, we observed some of the interesting patterns in the identified errors. As shown in Table 3, almost 70% of the identified *Type* 1 errors are actually gold errors. It supports our hypothesis that parsers agree only if an annotation is consistent enough in the treebank. Thus, it favors the ensembling of parsers for error detection. Another interesting fact is that some of the false-positives in *Type* 1 errors are very close cases. It is observed that even for experts, some of these cases are thought-provoking. Accurate annotation of these cases needs a proper understanding of the annotation guidelines and particularly, a deeper understanding of the language under study. One of the major issues in the creation of a treebank for any language is evaluating annotator's understanding and training them accordingly with the critical cases as per annotation guidelines. We observed that most of the false-positive cases of *Type* 1 are in fact critical or tough annotation cases. Thus these cases can be utilized to train or evaluate annotators. In case of *Type* 2 errors the number of actual gold errors are not high. The observation of false-positives of *Type* 2 errors makes it clear that the validation time taken for these cases is low as experts can easily ignore the unintuitive errors detected. However, the number of gold errors in *Type* 2 may vary for corpora, based on the annotator's understanding. Most false-positives in *Type* 2 resulted from data sparsity. This type of errors affect the precision of the system badly. *Type* 3 and *Type* 4 errors are detected based on the attachment. Generally, in dependency annotation annotators are good at annotating attachments compared to label annotation. As expected, these two types contain lesser number of gold errors compared to *Type* 1, but still in some inconsistent cases they identified good cases

of possible errors.

6 Discussion and Future Work

We have explored a different approach compared to previous efforts in error detection in annotated corpora. The approach is language independent. We have reported results on the Hindi and Urdu treebanks. The results (Recall) of our system on these treebanks are 73.42% and 81.76% respectively as shown in Table 3. Results reported by previous efforts using statistical approach on the Hindi treebank using huge training data (290k) is 73.12%. After combining with rule based system they are able to achieve 81.49% as in [1]. Our results are comparable to statistical system performance in spite of low training data in our case. The system outperformed in case of the Urdu treebank inspite of having relatively lesser training data (160k) compared to the Hindi training data (290k) used in the earlier approach. We detected errors across the treebank and then for evaluation we picked errorneous cases randomly from the entire treebank whereas in case of previous efforts all the evaluation results are reported on separate training data and testing data, thus results reported by our system are consistent compared to previous methods. *Type* 1 cases detected by the system can be utilized to constantly improve the annotation guidelines and to resolve some of the ambiguities. The system is limited by the available state of the art parsers. Apart from the detection of inconsistencies in the treebanks, we also prioritized them based on the confidence scores, which helps experts to validate errors more effectively. Data sparsity problem is a limitation due to which parsers are unable to learn the consistencies in the treebanks widely. Our future work includes detecting these cases using the parsers confidence score effectively. Our future work also includes handling sparsity issue by considering coarse label set at higher granular level. We also plan to embed the rule based system which uses robust rules specific to annotation guidelines and to a language as referred in [1].

7 Conclusion

We proposed a fully automated and a language independent approach to detect inconsistencies in dependency treebanks. The proposed system is able to detect 81.76% of existing errors in Urdu treebank and 72.64 % of existing results in a data which is part of the Hindi treebank. The inherent limitations of the system proposed are the infrequent cases. We proposed the prioritization of errors identified by our system which further enhances the validation process. The detected errors are also categorized apart from prioritization which helps experts in the process of validation. The use of confidence scores to detect inconsistencies in a treebank will be the focus of our future work.

References

[1] Rahul Agarwal, Bharat Ram Ambati, and Dipti Misra Sharma. A Hybrid Approach to Error Detection in a Treebank and its Impact on Manual Validation Time. In *Linguistic Issues in Language Technology*, volume 7, 2012.

[2] Bhasha Agrawal, Rahul Agarwal, Samar Husain, and Dipti M Sharma. An automatic approach to treebank error detection using a dependency parser. In *Computational Linguistics and Intelligent Text Processing*, pages 294–303. Springer, 2013.

[3] Bharat Ram Ambati, Rahul Agarwal, Mridul Gupta, Samar Husain, and Dipti Misra Sharma. Error detection for Treebank Validation. In *The 9th International workshop on Asian Language Resources (ALR). Chiang Mai, Thailand*, 2011.

[4] Bharat Ram Ambati, Phani Gadde, and Karan Jindal. Experiments in Indian language dependency parsing. *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, pages 32–37, 2009.

[5] Bharat Ram Ambati, Mridul Gupta, Samar Husain, and Dipti Misra Sharma. A High Recall Error Identification Tool for Hindi Treebank Validation. In *LREC*, 2010.

[6] Rafiya Begum, Samar Husain, Arun Dhwaj, Dipti Misra Sharma, Lakshmi Bai, and Rajeev Sangal. Dependency Annotation Scheme for Indian Languages. In *IJCNLP*, pages 721–726, 2008.

[7] A. Bharati, D.M. Sharma, S. Husain, L. Bai, R. Begum, and R. Sangal. AnnCorra: TreeBanks for Indian Languages Guidelines for Annotating Hindi TreeBank (version–2.0), 2009.

[8] Riyaz Ahmad Bhat, Sambhav Jain, and Dipti Misra Sharma. Experiments on Dependency Parsing of Urdu. In *Proceedings of The 11th International Workshop on Treebanks and Linguistic Theories (TLT11)*, 2012.

[9] Riyaz Ahmad Bhat and Dipti Misra Sharma. A dependency treebank of urdu and its evaluation. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 157–165. Association for Computational Linguistics, 2012.

[10] Riyaz Ahmad Bhat and Dipti Misra Sharma. Non-Projective Structures in Indian Language Treebanks. In *Proceedings of The 11th International Workshop on Treebanks and Linguistic Theories (TLT11)*, 2012.

[11] Rajesh Bhatt, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Misra Sharma, and Fei Xia. A multi-representational and multi-layered treebank for Hindi/Urdu. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 186–189. Association for Computational Linguistics, 2009.

[12] Daniël De Kok, Jianqiang Ma, and Gertjan Van Noord. A generalized method for iterative error mining in parsing results. In *Proceedings of the 2009 Workshop on Grammar Engineering Across Frameworks*, pages 71–79. Association for Computational Linguistics, 2009.

[13] Markus Dickinson and W Detmar Meurers. Detecting inconsistencies in treebanks. In *Proceedings of TLT*, volume 3, pages 45–56, 2003.

[14] Markus Dickinson and W Detmar Meurers. Detecting errors in discontinuous structural annotation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 322–329. Association for Computational Linguistics, 2005.

[15] Eleazar Eskin. Detecting errors within a corpus using anomaly detection. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 148–153. Association for Computational Linguistics, 2000.

[16] Johan Hall, Jens Nilsson, and Joakim Nivre. Single malt or blended? A study in multilingual parser optimization. In *Trends in Parsing Technology*, pages 19–33. Springer, 2010.

[17] Sambhav Jain and Bhasha Agrawal. A Dynamic Confusion Score for Dependency Arc Labels. In *Proceedings of The 6th International Joint Conference on Natural Language Processing (IJCNLP)*, 2013.

[18] Kaarel Kaljurand. Checking treebank consistency to find annotation errors, 2004.

[19] Puneeth Kukkadapu, Deepak Kumar Malladi, and Aswarth Dara. Ensembling Various Dependency Parsers: Adopting Turbo Parser for Indian Languages. In *24th International Conference on Computational Linguistics*, page 179, 2012.

[20] André FT Martins, Noah A Smith, Eric P Xing, Pedro MQ Aguiar, and Mário AT Figueiredo. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44. Association for Computational Linguistics, 2010.

[21] Avihai Mejer and Koby Crammer. Are you sure?: confidence in prediction of dependency tree edges. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 573–576. Association for Computational Linguistics, 2012.

[22] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

[23] Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 221–225. Association for Computational Linguistics, 2006.

[24] Karan Singla, Aniruddha Tammewar, Naman Jain, and Sambhav Jain. Two-stage Approach for Hindi Dependency Parsing Using MaltParser. *Training*, 12041(268,093):22–27, 2012.

[25] Hans van Halteren. The detection of inconsistency in manually tagged text. In *Proceedings of LINC2000*, 2000.

[26] Alexander Volokh and Günter Neumann. Automatic Detection and Correction of Errors in Dependency Treebanks. In *ACL (Short Papers)*, pages 346–350, 2011.

Repurposing Treebanks

Archna Bhatia,* Michael Deeringer,† Matthew Gardner,* Carlos Ramírez,*
Lori Levin,* Owen Rambow†

*LTI
CMU
Pittsburgh, PA, USA

†CCLS
Columbia University
New York, NY, USA

E-mail: archnab@cs.cmu.edu

Abstract

This paper describes a methodology for efficient revision and reformatting of treebanks. The basic unit of revision is a local subtree (a parent node and its immediate children). The treebank is first broken down into a list of unique types of local subtrees and then the head daughter is specified for each unique type. We tested the approach on the English PTB. We believe that this type-based approach is an efficient way to annotate head daughters and dependencies. In the course of doing this work, we have produced an annotation tool for identifying head daughters in local sub-trees. We have also produced a gold standard for head selection that covers 90% of tokens. The gold standard can be used for training head rules, and can easily be revised for trying alternate approaches to headedness in the treebank.

1 Introduction

The attendees at this conference are all aware that treebanking is a labor-intensive process even when semi-automated. Consequently, a treebank may have a long life span without change. However, trends in linguistics and language technologies are constantly changing, and it may be desirable to rejuvenate a treebank without having to completely rebuild it from scratch.

Interoperability is a recent trend that calls for large scale revision of treebanks. It refers to the use of common tag sets such as part of speech tags and dependency labels as well as common treatments of constructions such as coordinate structures. Interoperability is desirable for computational processing of treebanks.

Linguistically oriented projects have fostered interoperability under the guise of a linguistic theory. The Prague treebanks are interoperable with each other based on the Prague School of linguistics. Pargram [12] is another project that has resulted in interoperable treebanks and grammars based on Lexical Functional Grammar. However, treebanks from one linguistic theory are not necessarily in-

teroperable with another. What if we want to start with diverse treebanks from different research groups and make them interoperable?

Two recent efforts on interoperability are the Prague HamleDT [15] project and Google's universal dependency labels for cross-linguistic transfer parsing project [10]. Tsarfaty [13] has also proposed a universal set of dependency labels but also allows the dependency labels to have sub-types to allow for typological variation. HamleDT has proposed standards for several constructions such as coordinate structures [15, 11] and has interoperationalized 29 treebanks from other projects [15].

The project we are reporting on in this paper is called "Syntax based on a Universal Predicate-Argument Representation" (SUPA). It is also concerned with interoperability in treatment of dependency labels and constructions, and also hopes to eventually convert many treebanks with different theoretical frameworks into SUPA, a dependency representation, and hence make them interoperable. However, we would like to suggest that interoperability will not be a one time event, but may happen many times in the life of a treebank. We do not believe that there is any one tagset or treatment of constructions that is optimal for every computational application that employs treebanks.

This paper describes one experiment in treebank conversion, conducted within the SUPA project. It involves a conversion of the Penn English Treebank (PTB) to the SUPA dependency format. We did not follow the standard approach of using complete sentences only as test suites for conversion from phrase structure to dependency format. Instead we followed a type-based approach to create test suites for, e.g. head selection (where each type is a unique local subtree found in PTB), and making a gold standard for them for testing during the conversion from phrase structure to dependency format. Our type based approach was originally intended to contribute to a more careful formulation of head rules [7, 3, 5], as we had found during the conversion process that the head rules used earlier did not necessarily always select the correct heads (in fact there were quite many mismatches which a thorough linguistic investigation would easily point out). In the process of fixing the head rules so they represent English better, we devised the type-based approach used for conversion and testing which we are discussing in this paper. In the following sections we will describe the SUPA dependency format, summarize some statistics about unique local subtrees in PTB, and present a tool that supports a type-based approach for marking headedness.

2 SUPA Dependency Format

In this section we summarize the choices we have made for interoperability in the SUPA project. However, the reader should keep in mind that we intend to be flexible about treebank revisions allowing multiple approaches even within SUPA.

2.1 Functional and Lexical Heads

In linguistic theory, function words and lexical words work together to head phrases [1, 2]. However, dependency tree formats do not typically allow for co-headed phrases. Therefore, it is necessary to choose the lexical head, the functional head, or make a single token out of a combination of lexical items, which contain both lexical and functional heads, such as a verb complex like *has been writing*. We follow Stanford Dependency Parser [6] in the belief that functional and lexical headed dependencies are both valid and that there should be multiple views of co-headed constructions.

In the experiment described in this paper, we have chosen a more limited approach and have made specific decisions about functional and lexical categories in English: in general the lexical categories are heads, however, auxiliary verbs, when present, are taken as heads; prepositions are taken as heads; the other functional categories are not taken as heads. Hence complementizers are not heads; and similarly determiners are not taken as heads for this experiment. The object of this paper is not to defend this particular choice, and in principle nothing in this paper hinges on this choice of heads. There are many flat structures in PTB where headedness is unclear, as we will describe below. In such cases, we have selected one of the possible options for the purpose of conversion to SUPA dependency.

2.2 Coordinate Structures

As described by [15], there are several ways to represent coordinate structures in dependency trees. For the experiment described in this paper we have chosen the first conjunct as head. The conjunction word is a dependent of the first conjunct, and the second conjunct is dependent on the conjunction. In our SUPA version of PTB, we pre-process coordinate structures so that [XP A & B] is reanalyzed as [XP A [CONJPP & B]]. In case of multiple conjunctions, such as [A & B & C] the coordinated construction is reanalyzed as [XP A [CONJPP & [YP B [CONJPP & C]]]]. Figure 1 shows an example of a coordinate structure for an NP from PTB.

```
(NP
 (NP (DT the) (NNP National)
     (NNP Cancer) (NNP Institute))
 (CC and)
 (NP
   (NP (DT the) (JJ medical) (NNS schools))
   (PP (IN of)
     (NP
       (NP (NNP Harvard) (NNP University))
       (CC and)
       (NP (NNP Boston) (NNP University))))))
```

Figure 1: Constituency Tree for a coordinated NP from the Penn English Treebank

Figure 2 shows our dependency representation of the above coordinate structure in the CONLL format which is the result of conversion from the output of preprocessing the original coordinate structure to have the above-mentioned reanalysis. Please note that "???" in column 8 in figure 2 as well as in all the following figures representing SUPA will eventually be replaced by the dependency labels, which are not the topic of this paper. Currently everything other than Subj, Obj-DO, Obj-IO and ROOT is expressed using "???". The symbol "-" in columns 3, 4, 6, 9 and 10 is used to fill the slots for information that we were not concerned with for this part of the project, however, to keep the CoNLL format, we maintained the slots using "-" as a filler. These can be replaced with corresponding information when needed.

```
1     the          _     _     DT     _     4     ???     _     _     Institute
2     National     _     _     NNP    _     4     ???     _     _     Institute
3     Cancer       _     _     NNP    _     4     ???     _     _     Institute
4     Institute    _     _     NNP    _     0     ROOT    _     _     ROOT
5     and          _     _     CC     _     4     ???     _     _     Institute
6     the          _     _     DT     _     8     ???     _     _     schools
7     medical      _     _     JJ     _     8     ???     _     _     schools
8     schools      _     _     NNS    _     5     ???     _     _     and
9     of           _     _     IN     _     8     ???     _     _     schools
10    Harvard      _     _     NNP    _     11    ???     _     _     University
11    University   _     _     NNP    _     9     ???     _     _     of
12    and          _     _     CC     _     11    ???     _     _     University
13    Boston       _     _     NNP    _     14    ???     _     _     University
14    University   _     _     NNP    _     12    ???     _     _     and
```

Figure 2: Dependency representation for the coordinated NP above in CONLL format

2.3 Long Distance Dependencies

In our dependency trees, words that are displaced by long distance dependencies and are presented as antecedents of traces or other empty categories in the PTB are represented in their “deep structure” positions.

See figure 3 for an example of tree involving a long distance dependency and figure 4 for its corresponding dependency representation in the CONLL format.

3 Type-Based Conversion of PTB

3.1 Extraction of Unique Local Subtrees

In this section we describe the process for type based treebank conversion. From the training section of the PTB, we extracted all the unique subtrees of depth two (i.e. the parent and child nodes). These were grouped according to the parent node

```
(SBAR-NOM-PRD                          (SBAR-NOM-PRD
  (ADVP (RB just))                       (ADVP (RB just))
  (WHNP-2 (WP what))
  (S                                     (S
    (NP-SBJ (PRP we))                      (NP-SBJ (PRP we))
    (VP (VBP are)                          (VP (VBP are)
      (VP (VBG stumbling)                      (VP (VBG stumbling)
        (PP-CLR (IN over)                          (PP-CLR (IN over)
          (NP (-NONE- *T*-2)))))))                   (NP (WHNP
                                                           (WP what)))))))))
```

Figure 3: Original constituency tree involving a long distance dependency from the Penn English Treebank (left), tree after pre-processing steps (right)

```
1     just      _     _     RB     _     4     ???     _     _     are
2     what      _     _     WP     _     6     ???     _     _     over
3     we        _     _     PRP    _     4     SUBJ    _     _     are
4     are       _     _     VBP    _     0     ROOT    _     _     ROOT
5     stumbling _     _     VBG    _     4     ???     _     _     are
6     over      _     _     IN     _     5     ???     _     _     stumbling
```

Figure 4: Dependency representation for the constituency tree involving a long distance dependency above in CONLL format

(e.g. NP, VP, PP). (We excluded preterminal-terminal subtrees.) We used each of these groupings as the test suite for that node. We call the members of the test suite "patterns". We created these test suites to be able to test our head rules, as well as dependency relations against these test suites. There are 26 distinct test suites. For each pattern, we chose one full subtree from the corpus (including all terminals) to represent it: these are subtrees whose root node is expanded using the pattern. Figure 5 shows five full subtrees from the ADJP test suite. For example, the first tree in figure 5 below represents the pattern type: (ADJP JJ), the second tree represents the pattern: (ADJP RB JJ), and so on. Note that if the type based approach is justified, any example would be representative of the type. We provide a confirmation of this claim in section 5.

The examples were entered in our test suite in their full form as well as in their simplified form. For creating the simplified forms, we came up with a set of heuristics to prune adjuncts and other unnecessary nodes, while preserving enough of the tree so that it is interpretable and grammatical. For example, *but to sell at a week price, even if it means losing some steel-related tax-loss carryforwards* was simplified into *but to sell*.

As is expected, the test suites varied considerably in the number of types of patterns they contained, from just 5 types for the WHPP test suite to 6498 types for the NP test suite. The types in these test suites were organized according to their token frequencies, beginning with the type with the highest token count to types

with the lower token counts.

```
(ADJP-PRD (JJ curly))

(ADJP-PRD (RB unusually) (JJ resilient))

(ADJP-PRD (JJ aware)
  (PP (IN of)
    (NP
      (NP (DT any) (NN research))
      (PP (IN on)
        (NP
          (NP (NNS smokers))
          (PP (IN of)
            (NP (DT the) (NNP Kent) (NNS cigarettes))))))))

(ADJP (CD 83.4) (NN %))

(ADJP
  (QP ($ $) (CD 2.29) (CD billion))
  (-NONE- *U*))
```

Figure 5: A part of the ADJP test suite extracted from Penn English Treebank

3.2 The Type-Based Annotation Tool

Since one of our goals was to create a gold standard for head selection for these test suites, we designed a tool that makes it convenient to create such a gold standard. For the gold standard, we decided to cover as many types so as to reach a minimum of 90% of token coverage in each of the test suites. There are 2 parts to the tool: (i) The summary page provides the information regarding the number and percentage of the types as well as tokens covered as we annotate the data for the correct headedness for each pattern type. It also provides other statistics that we found to be useful for our purposes. Figure 6 below presents a snapshot of a part of the summary page from the tool. (ii) The test suite pages provide a way to look at the pattern type, its token frequency, the original constituency tree, the constituency tree after pre-processing of coordinate structures and long distance dependencies, the simplified constituency tree, and the current dependency representations corresponding to the original constituency tree as well as the simplified constituency tree. The test suite pages provide a way for the annotators to select the correct head and state whether the dependency representation correctly identified the head or not, etc. Figure 7 presents a snapshot of a part of the test suite VP.

3.3 The Gold Standard

We used the annotation tool to create a gold standard for head selection. In the test suite for each parent node (NP, S, SBAR, etc), we started with the most frequent

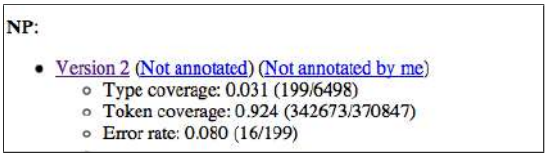


Figure 6: Summary of the NP test suite head annotation information from the tool

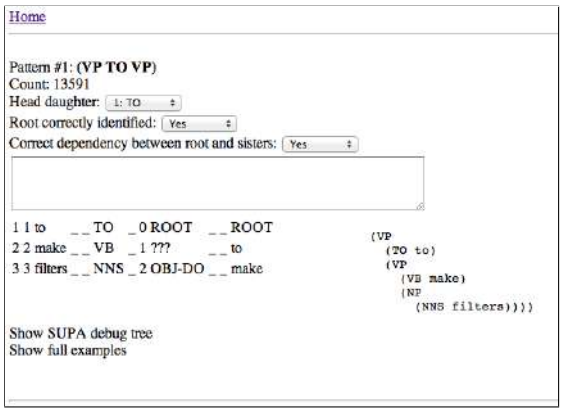


Figure 7: A part of the VP test suite from the tool

subtree types and kept going until we had annotated enough types to cover 90% of the tokens of subtrees headed by that parent node. We calculated inter annotator agreement for 215 randomly selected pattern types from the pool of all the pattern types (all patterns from all the test suites put together) which roughly represented the distribution of the test suite categories (e.g. NP, VP, S etc) in the PTB training set. We report a very high agreement with Cohen's Kappa = 0.86. The confusion matrix is shown in Figure 8 below. We are happy to share our gold standard. However, we expect to change it frequently using the annotation tool in order to experiment with different approaches to headedness and their effect on applications in parsing and machine translation.

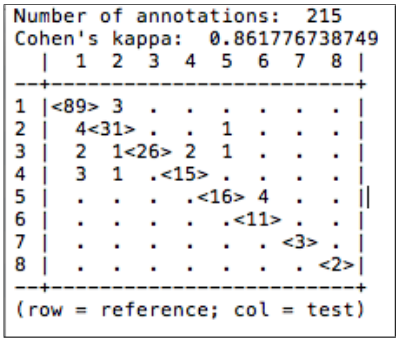


Figure 8: Inter annotator agreement and the confusion matrix for head annotations

In producing the gold standard, we found certain systematic and certain unrelated errors in the PTB in terms of use of tags etc. We have decided to not correct these errors and provide an annotation taking the input as it is in the PTB for this version of gold standard. However, in the future, we would like to have another version of the gold standard that would represent the corrected PTB. Besides these errors, we also came up with certain hard cases. They were hard in the sense that when we applied the diagnostics (such as X-bar theory of headedness that the head projects to the higher level, distributivity) to determine headedness, different diagnostics gave different or opposite results. Hence we had to make a decision regarding the analysis of a construction and the preference for a certain diagnostic for that construction. Below are some examples.

Ambiguity in Coordination constructions: Many coordination constructions were ambiguous as PTB did not provide an internal analysis of coordination, instead the flat structure was provided. In such cases, as mentioned above as well, we had to enforce a structure so as to be able to convert to dependency (note dependency requires a head for each phrase to express relationships between it and other constituents in the phrase). For example, *[construction and property management]* could be interpreted as *[[construction and property] management]* or as *[construction and [property management]]*. In absence of the structure in PTB and any other strong evidence against the following, we defaulted to *[[construction and property] management]* where the coordinated phrase is assumed to modify *management* or it is assumed to form a compound noun with it; and the *management* is taken as the head of the whole phrase.

Difficulty in applying headedness diagnostics in QPs: As mentioned above, there were certain cases where it was hard to apply the headedness diagnostics we employed in general to determine the head in an ambiguous situation. For example, the QP test suite consisted of QPs containing JJS, IN, RB, CD etc. If we look at these phrases, we find that structurally they look like ADJPs (e.g. *fewer than 100* [QP JJR IN CD]) or PPs (e.g. *about 160* [QP IN CD]) or AdvPs (e.g. *almost half* [QP RB DT]). However distributionally they appear in the same position as CDs. For example, note *fewer than 100* in the following phrase appears in the slot where a CD normally appears: *the* **fewer than 100** *smart students*. Thus we found a conflict in headedness properties here. For QPs, we decided to go ahead with the distributional criterion and hence select the CD as the head.

Difficulty in determining a head due to the test suite being a non homogenous group of constructions: Besides the QPs, we found some other test suites, e.g., FRAG and X also to be quite problematic. The reason for this was that these categories seemed to have many non homogenous constructions. Hence it was hard to have a consistent analysis for the patterns in these categories. These categories have caused problems for parsing due to their non homogenous nature. They do not have a consistent analysis. For example, lists sometimes get treated as appositives or nested noun phrases etc. There are attempts in the literature at getting a better analysis for such cases, see [8].

Difficulty in selecting a head in a Multi-Word Expression: Also it was hard

to determine the head in multi-word expressions (MWEs). For example, for ADVP *at least* [ADVP IN JJS], should IN be the head or JJS? The head diagnostics do not help in choosing one over the other. Here are a few more such cases: [CONJP RB RB IN], e.g. *as well as*; [CONJP RB RB], e.g. *not only*; [CONJP CC RB], e.g. *but also*. Hence we see that for MWEs, it was hard to determine the head as all the parts seemed to contribute equally. In such cases, we either decided to pick the rightmost or the leftmost element depending on the category of the phrase.

4 Statistics about Types in PTB

In order for the type based approach to treebank revision to be useful it must be the case that few types account for a large proportion of tokens. Figure 9 is a type-token curve that shows that this is the case for noun phrases in PTB. However, some categories in PTB are less uniform. Figure 10 shows the type-token curve for the category FRAG, which contains a variety of unrelated constructions.

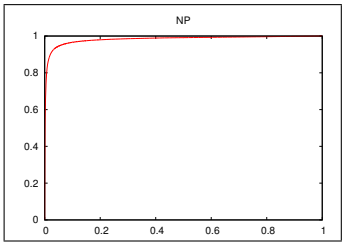


Figure 9: The token-type coverage in the NP test suite; the x-axis is the percentage of types, while the y-axis is the percentage of tokens

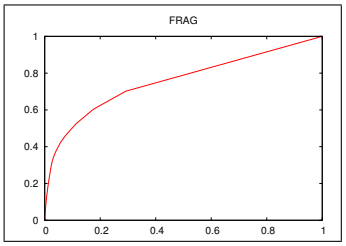


Figure 10: The token-type coverage in the FRAG test suite; the x-axis is the percentage of types, while the y-axis is the percentage of tokens

It is also important to verify how many complete trees consist only of the most frequent types of subtrees. Or to put it another way, how many trees contain at least one infrequent type of local subtree. Figure 11 shows that around 62% of trees consist only of subtrees that are in the top 20% of most frequent local subtree types. As a sanity check for the type-based approach (where we ordered types according to their frequency), we also examined the coverage of complete trees if

the types covered are randomly selected rather than based on their frequency. We found that to be able to cover 60% of the complete trees, we had to cover more than 90% of the randomly selected types. This suggests that ordering types by frequency has an advantage while revising the treebanks.

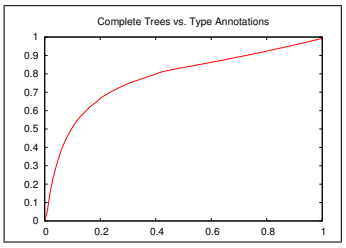


Figure 11: The token-type coverage in completely annotated trees; the x-axis shows the percentage of top trees annotated, and the y-axis shows the corresponding percentage of complete trees covered

5 Confirmation of the Type-Based Approach

We intend to use our gold standard for rapid development, testing, and revision of head rules. For this paper, however, we report on a much simpler experiment. We used the gold standard as a lookup table for a head selection task. The input to the task is PTB. For each tree, we look up each of its local subtrees in the gold standard and select the head daughter indicated in gold standard. The output is a PTB tree with the label -HEAD added to each head daughter. For example, (NP (DET the) (NN book)) is transformed into (NP (DET the) (NN-HEAD book)).

We conducted this experiment as a sanity check on whether one token is a good representative of a whole type. If there are many examples in the gold standard where the example we chose is not representative of all the tokens of that type, the gold standard will produce many wrong results when used as a lookup table.

Recall that our gold standard covers enough types for 90% token coverage at the local subtree level, but the number of complete trees consisting only of gold standard types is considerably less. We conducted our sanity check on the subset of trees that were completely covered by the gold standard. We randomly selected 200 sentences from that subset and used the gold standard as a table lookup for head selection as described above. We hand-checked each -HEAD label and found them all to be correct. As a follow up to the reviewers' comments about creating a gold standard beforehand instead of hand checking the head selection later on, we conducted another experiment. For this, we selected 200 random trees without any condition (i.e. without the condition that all local subtrees in the selected tree are covered by the head table) and from previously unseen data, the development set of PTB. We created a gold standard by manually marking "-HEAD" on each head of each local subtree. We then checked how many of these local subtrees

were covered by the head table. Also we checked how many of the covered local subtrees were marked accurately by head selection using the head table. We found that our head table was able to cover 83% of the local subtrees from this unseen data (note the head table was created using the types found in the training set of the PTB). With respect to accuracy, we did not find any cases where the head table selected an incorrect head, unless the head table entries themselves were not correct. We did find a couple of head table entries which were not consistent with our general approach of head selection, e.g. whether *'s* should be the head or the possessor nominal. We fixed those in the head table.

6 Conclusion and comparison with other approaches

We have produced a tool for annotation of head selection in local subtrees and have applied it to PTB training set to produce a gold standard consisting of a set of unique local subtree types and the correct head daughter for each type. We have conducted sanity check experiments to verify that one example of each type is adequate for developing head marking rules, and that annotating higher frequency types only does cover a large portion of PTB data.

However, our task was not painless. As mentioned above, we did not correct errors in PTB and therefore had to deal with many errors in part of speech tags and many constituent structures that we did not agree with. Categories like FRAG are very diverse and there is probably no correct generalization about headedness. Furthermore, many headedness decisions are not straightforward as we discussed in section on Gold Standard above.

Conversion from phrase structure to dependency format [14, 9] is frequently facilitated by head rules [7, 3, 5]. Head rules are a brilliant invention but their use in phrase structure to dependency conversion is a fix to an artificial situation: some phrase structure treebanks do not strictly follow X-bar theory (or other phrase structure theory of headedness) and they had to be converted to dependency structure, which does not allow for unheaded (exocentric) constructions. In this strange circumstance, head rules must be designed to assign reasonable heads to phrases where the original treebankers may not have even intended the phrase to be headed. Some head rules are accurate, others are heuristic, and sometimes they use a default for unanticipated situations.

In a Magerman/Hwa-style head rule system [5], there is one head rule for each type of parent node in the phrase structure treebank. The rule shown in figure 12 is for the category NP. The head rule specifies a partial ordering of all possible daughter nodes. The partial orderings in the rule below are (NX PRP), (NNS NN NNP NNPS), and so on. "r" and "l" indicate rightmost or leftmost. The "r" in (r NNS NN NNP NNPS) indicates that the rightmost noun-like item is taken to be the head, which is generally the case for compounds in English (e.g., *lung cancer cure*). The notation "(r)" at the end of the rule indicates that if none of the previous categories match, the default is the rightmost daughter of NP. (l NP) is for cases

where a PP is adjoined to the right of an NP.

```
NP (r NX PRP) (r NNS NN NNP NNPS) (l NP) (r CD FW SBAR RBS) (r)
```

Figure 12: An example of a Hwa style headrule

This type of head rule assumes that all of the potential daughters can be partially ordered in their likelihood to be the head. The head rules are problematic when the ordering depends on specific lexical items, a condition that is exacerbated by non-optimal part of speech labels in PTB. In PTB, *such* and *enough* are both JJ, and *such tiny* and *tiny enough* are both instances of (ADJP JJ JJ). Head rules can get only one of them right. Our table-lookup method for head selection does not capture generalizations about word order and direction of headedness, but it captures the specific context around the head. Next we plan to combine both approaches to head selection. Thus, first, the algorithm looks for the pattern type in the head table for head selection. However in the absence of the type in the head table, it falls back on the head rules to determine what the head should be in a local subtree. This way we derive benefits of both the approaches: the head table allows capturing the specific surrounding context, and in cases where it is not useful (due to a missing pattern in the head table as is expected since we might not want to spend resources on covering each pattern type, e.g. the singleton cases) the generalization feature of the head rules can come handy. Our future experiments include looking for improvements in parsing with improved head rules and experiments on how dependency label sets (universal vs language specific) affect monolingual and cross-lingual parser training.

Acknowledgments

We thank anonymous reviewers for useful feedback. This work was supported by a Google Faculty Award. Deeringer is now at Google, Inc.

References

[1] J. Bresnan. *Lexical-functional syntax*. Blackwell, 2001.

[2] N. Chomsky. *The minimalist program*. MIT Press, 1995.

[3] M. Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4), 1999.

[4] D. Gerdts. Mapping halkomelem grammatical relations. *Linguistics*, 31, 1993.

[5] R. Hwa, P. Resnik, A. Weinberg, C. I. Cabezas, and O. Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(3), 2005.

[6] D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 2003.

[7] D. M. Magerman. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 1995.

[8] M. De Marneffe, M. Connor, N. Silveira, S. R. Bowman, T. Dozat, and C. D. Manning. More Constructions, More Genres: Extending Stanford Dependencies. In *Proceedings of the Second International Conference on Dependency Linguistics*, 2013.

[9] M. De Marneffe, B. Maccartney, and C. D. Manning. Generating typed dependency parses from phrase structure parses. In *LREC*, 2006.

[10] R. McDonald, J. Nivre, Y. Quirmbach-Brundage, Y. Goldberg, D. Das, K. Ganchev, K. Hall, S. Petrov, H. Zhang, O. Täckström, C. Bedini, N. B. Castelló, and J. Lee. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

[11] M. Popel, D. Mareček, J. Štěpánek, D. Zeman, and Z. Žabokrtský. Coordination structures in dependency treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

[12] S. Sulger, M. Butt, T. H. King, P. Meurer, T. Laczkó, G. Rákosi, C. B. Dione, H. Dyvik, V. Rosén, K. De Smedt, A. Patejuk, O. Cetinoğlu, I. W. Arka, and M. Mistica. Pargrambank: The Pargram Parallel Treebank. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

[13] R. Tsarfaty. A unified morpho-syntactic scheme of Stanford Dependencies. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013.

[14] F. Xia and M. Palmer. Converting dependency structures to phrase structures. In *Proceedings of the first international conference on Human language technology research*, 2001.

[15] D. Zeman, D. Mareček, M. Popel, L. Ramasamy, J. Štěpánek, Z. Žabokrtský, and J. Hajič. Hamledt: To parse or not to parse? In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, 2012.

Reliability and Meta-reliability of Language Resources: Ready to initiate the Integrity Debate?

António Branco

University of Lisbon
Antonio.Branco@di.fc.ul.pt

1 Introduction

Given the increasing complexity and expertise involved in the development of language resources, there have been a growing interest in finding mechanisms so that the designing and the development of language resources may be taken as a first class citizen in terms of scientific work, and accordingly cvs and careers of individual researchers can be fairly credited and rewarded for that. Ongoing initiatives such as the international standard language resource number (Choukri, 2013), or the studies on metrics to ascertain the reliability of linguistically interpreted data sets (e.g. Artstein and Poesio, 2008) are just a few illustrative examples of this trend.

Concomitant to this movement of reinforced scientific credibility, but in an opposite direction, there have been appearing worrying signs that, in what concerns mature and well established scientific fields, scientific activities and results may be untrustable to an extent larger than possibly expected and acceptable. That this issue has recently hit the mass media[1] is but an indicator of the volume and relevance of these signs, whose assessment and discussion became unavoidable across all sectors of the international scientific system.

These signs have been related, for instance, to the realization that for a considerable proportion of published results their replication is not being obtained by independent researchers (e.g. Florian et al., 2011; Belgley and Ellis, 2012); to the deliberately falsified submissions of papers for publication, with fabricated errors and fake authors, which get easily accepted even in respectable journals (Bohannon, 2013); or to the outcome of inquiries to scientists on questionable practices, with scores higher than one

[1] Unreliable Research: Trouble at the Lab, *The Economist*, October 19th, 2013.
Hiltzik, Michael, 2013, Science has Lost its Way, at a big Cost to Humanity, *Los Angeles Times*, October 17, 2013.
Zimmer, Carl, 2012, A Sharp Rise in Retractions Prompts Calls for Reform, *The New York Times,* April 16, 2012
Begley, Sharon, 2012, In Cancer Science, Many "Discoveries" don't Hold up, *Reuters*, March 28th, 2012.
Nail, Gautam, 2011, Scientists' Elusive Goal: Reproducing Study Results, *The Wall Street Journal*, December 2, 2011.

might expect or would be ready to accept (Fanelli, 2009). For a receent and updated overview and further references on these signs, see (Stodden, 2013).

A number of causes have been aired for these state of affairs including, among others, increasingly sloppy reviewing; the growing number of so-called "minimal-threshold" journals; policies for publication that do not require the sharing of at least the raw or primary data; or the non disclosure of the software developed and used to obtain the results published. These causes have deserved serious scrutiny, including in the "World Conference on Research Integrity", whose third edition was held this year.[2]

Underneath these immediate causes, a number of factors have been pointed out, including, for instance, not enough negative incentives or peer-pressure to hamper the above practices; career and promotion pressure too biased for quantity; widespread disinterest on negative results as an intrinsic part of the scientific progress; widespread disfavoring of activities of replication by funding agencies; poor or non existent retraction procedures for results that are eventually noticed to be wrong or flawed after having been published; ideological pressure to get immediate financial return from research results; etc.

In Bill Frezza's bold opinion, the financial pressure on the scientific system "has created a moral hazard to scientific integrity no less threatening than the moral hazard to financial integrity that recently destroyed our banking system." (Frezza, 2011).

In the present invited talk at the 12th Workshop on Treebanks and Linguistic Theory, I am interested in contributing to initiate a debate on what part of the above issues may be recognized as having the conditions to be eventually happening also in our field, what part does not apply to it given its specific nature, and what may be the risks that may be specific to it. The ultimate goal of this exercise is to contribute for the reinforcement of the scientific credibility of language resources, and to the integrity of our scientific work around them.

Before proceeding, a word of clarification is in order, in particular to indicate what this talk is not about. It is not about what one might term as issues of empirical adequacy of linguistically interpreted data sets. These are issues related to the adequate interpretation of the markables. For instance, issues that occur if in the annotation of a corpus, as a result of a flawed design, the annotation principles or guidelines would wrongly require that what are standard grammatical prepositions be mistakenly annotated as adjectives, etc. These are the issues addressed, for instance, in (Zaenen, 2006).

It is not about issues of reliability of annotated data sets either. These are issues that are related to the adequate definition of the annotation methodology in view of minimizing errors in the application of the annotation guidelines, and that can be monitored by metrics involving inter-

[2] http://www.wcri2013.org

annotator agreement, etc. These are the issues addressed, for instance, in (Artstein and Paoesion, 2008).

This talk is about integrity issues that are associated to the overall scientific ecosystem where the development of language resources and the research around it takes place. These are issues related to the overall conditions that support the credibility of and trust on the scientific work and its results, and that remain to be addressed even if the issues on empirical adequacy and reliability of the data sets are eventually settled.

2 Essential replication

Let us take two key processes contributing for the integrity of scientific activity and results, reviewing and replication, the former applying before (or leading to) the publication of results, and the latter applying after these have been published. The point worth noting at this juncture is that the need for replication does not result from the fact the ideal of flawless reviewing cannot be attained. Replication is a first class citizen here and it is still needed to play a crucial role even in case flawless reviewing could have been ensured.

For the sake of concreteness, let us consider the example of natural sciences. And to keep the reflection at a general enough level, let us understand that to a large extent, advancements and discoveries reported in papers are obtained as a result of new ways on how to gather primary data and/or how to analyze them into secondary data and empirically supported generalizations. If an ideal flawless review process could be possible, for a top-scoring paper accepted for publication, in essence the reviewers would then have said ok to what? In essence, to what one would call, for lack of a better and more encompassing term, the "methodological" aspects reported in the paper.

For the sake of the point, let us leave intentional misconduct or fabrication of data aside. In spite of the existence of a correct methodology to collect the primary data, their actual gathering may have gone wrong as a consequence of some clerical error or some inadvertent practical slips. Likewise, the analysis into secondary data and generalizations may have not been appropriately executed also due to some fortuitous reasons. The point here is that, in general, for rich and complex enough data, reviewers have no means to detect this kind of problems, unless they would also run the experiments themselves and executed the analysis of the data. But that is what replication is all about, and for obvious practical reasons, it is not and cannot be under the scope of the reviewing process.

3 Emerging validation

How can these considerations be transposed to or help to think about the field of language resources, where the ultimate goal is the development of primary data (to be used at subsequent technological and scientific activities)? At a general enough level of appreciation, we should then start with the note that a typical research paper in our area focus in the first of the two parts indicated above. Though it tends to be seen as a positive feature that papers may report also on technological solutions or tools supported by the data set whose development is being reported, the key topic is clearly which methodological novelties are involved (e.g. a new bootstrapping approach, new languages involved, new relations between previously available data, etc.) and which data set, i.e. primary data, that these innovations have eventually led to.

Let us transpose the question above to our area: If an ideal flawless review process was possible, for a top-scoring paper accepted for publication, in essence the reviewers would then have said ok to what? Again, in essence, to the methodological aspects reported in the paper. And again, in spite of the existence of a correct methodology to collect the primary data, their actual gathering may have inadvertently gone wrong for a number of fortuitous reasons. And that is where and why "replication" has its key role to play.

And here we arrive at an important point of our discussion: what exactly can be, or should be understood as replication, or as playing the role of replication, in this research area of language resources?

Replication permits to go beyond the mere verification of the methodological issues by reviewers, as these are reported in successfully published papers. It permits to check if the execution of those methodological steps, procedures, calculations, processes, etc. actually lead to the results that are being reported. For the area of language resources, in a very narrow and strict sense, this might translate into redoing the data set whose development is reported in a given paper, which clearly is completely out of question for obvious practical reasons. In a less narrow and more sensible sense, this may translate into checking, even if only by an as smart sampling as possible, whether the data set that resulted is actually the one being announced in the paper.

As replication is different of and out of the scope of reviewing in natural sciences, also here in our area whatever the details of this validation process may be, it is not an assignment for reviewers. For one, because for a large number of papers on languages resources, the data sets whose development is being reported are not publicly made available by their authors. But even if they were, and in the growing number of those resources that are actually made available at the moment of the publication of the respective papers, it is obvious that reviewers have no practical conditions to proceed with such validation, which in the case of language resources may play the role analogous to the one replication plays in other areas.

As replication of experiments is a key element in the integrity of the scientific ecosystem of other sciences, validation of language resources cannot but be a key element for the integrity of scientific activities in our area.

4 Illusory impactfulness

It may be tempting to consider that in the case of language resources, their validation is eventually taken care of not at a specific moment or in some dedicated occasion or explicit procedure, but that this just happens implicitly by the "invisible hand" of the different impact of the different resources in the community of researchers and users. A given resource has a larger impact if it is used more frequently and referred to more often in a larger number of papers. But the level of impact of a resource illusoriously correlates with the possible level at which such resource had been validated, even if supposedly by the mere effect of the usage that the community is doing of it.

For languages for which there is a small community of researchers working on it, and little or no funding exists to do so, a resource referred to only a very few times may be a perfectly developed data set, in accordance to the respective methodological principles and guidelines, that may happen to be fully adequate in linguistic terms. The same holds for resources that support work on less researched topics, which comparatively may receive a very small number of references and yet be an extraordinarily well-developed resource, which would top score in any rigorous validation process.

In the opposite direction, it occurs also that a resource may have a widespread usage and receive a high number of references and yet its validation would indicate suboptimal scores (Van Halteren, 2000; Eskin, 2000; Dickinson and Meurers, 2003; Tylman and Simov, 2004; Dickinson and Meurers, 2005). It is enough, for instance, that it is the first of its kind for English and/or supports research in a very hot topic.

Current mainstream research on natural language processing is about getting increasingly better evaluation scores for the relevant type of tools or applications while working with some given data sets (to ensure comparability), which *ipso facto* become the de facto standard data sets. And this can be pursued, and is actually pursued, whether or not those data sets had been correctly developed or had been gone through any validation process.

As Annie Zaenen put it in a humorous way when discussing the specific case of the development of language resources annotated for coreference: "Of course, as long as the task is to provide material to develop and refine machine-learning techniques, much of this doesn't matter. Whether *Henry Higgins* and *Eliza Doolittle* are referring to the same entity or not is of no interest in that context. The technique has only to show that if it is told that

they are coreferent because they had the same job (even at different moments), then it can also learn that George Bush and his father are coreferent." (Zaenen, 2006, p.579).

5 Putting on the agenda

For other long-established scientific areas, the discussion on replication of experiments and other integrity aspects of the scientific work has definitely made its way into the public agenda on science. And the discussion on mechanisms, conditions and incentives to foster, support, fund or perform replication has arrived to stay.[3] By the same token, we should bring the discussion on the validation of resources to the agenda of our community, and add it to other possible forward-looking issues currently aimed at strengthening the conditions of our research work.

When considering the actual enormous amount of effort, time and perseverance that is necessary to put in place a large enough data set that may be annotated with some quite sophisticated linguistic interpretation, under some stringent reliability ensuring methodology, one has to admit that the effort and conditions needed to publish a paper reporting on its development or fill in its metadata record, and get credited for it, is incomparably much ligther. Validation is a crucial element to help preventing and diverting possibly unduly inflated or even void reporting.

The organizations, initiatives or platforms operating the distribution of language resources, such as ELDA, LDC, OLAC, META-SHARE or CLARIN among others, have been driving forces of a continuous endeavor to support and foster the research area of language resources. In my view, it is only natural that, in order for them to evolve along the natural progression of the field and its new demands, the community of researchers expects that the mission of these organizations be extended. In particular, we can expect that these organizations play a key role in contributing to research integrity by being independent stakeholders to whom the role of validating language resources is trusted.

It is certain that in their regular daily operation, the language resources distribution organizations have been proceeding with instrumental verification of the resources that they receive to be distributed, at least to check whether the content of the packages match the description of the

[3] The Reproducibility Initiative (www.scienceexchange.com/reproducibility) was launched in 2012 by several prominent scientific journals and organizations in response to revelations from the pharmaceutical industry that a large proportion of published cancer research cannot be reproduced. It intends to identify and reward high quality reproducible research through independent validation of key experimental results.
The Center for Open Science (centerforopenscience.org) is a non-profit organization founded in January 2013 to increase openness, integrity, and reproducibility of scientific research.

resource provided in its metadata record.[4] But given the discussion above, the point is that this may need to be re-addressed under an entirely new light, and under renewed conditions. More than being just an internal procedure, validation of language resources plays a unique role in the whole ecosystem where the research work on and around language may strive, raises its profile, and hope to keep progressing according to the highest scientific standards.

How the distribution organizations may fulfill this role, assume this responsibility and make a key contribution for the progress of the area is a much-needed debate, which should welcome different views from different actors, and which the present paper would like to trigger. I would though dare to venture that at least a couple of ingredients will be crucial: the validation of language resources should be systematic and public.

For different types of datasets, practically feasible and *de facto* standard procedures should emerge on how to proceed with their validation.

And, as a way of a badge of validity, the metadata record of each resource should publicly indicate which kind of validation procedure it was submitted to, and what were the scores obtained for the different validation parameters if applicable.

Clearly, this will bring the language resources distribution organizations from the level of being instrumental supporters to be key players in the sustainability of the whole area, representing and ensuring a much needed self-regulatory endeavor of the scientific community they were aimed to serve when they were initially set up.

References

Unreliable Research: Trouble at the Lab, *The Economist*, October 19, 2013, online edition. http://www.economist.com/news/briefing/21588057-scientists-think-science-self-correcting-alarming-degree-it-not-trouble

Artstein, Ron, and Maximo Poesio, 2008, Inter-Coder Agreement for Computational Linguistics, *Computational Linguistics*, 34(4), pp.555-596.

Begley, Sharon, 2012, In cancer science, many "discoveries" don't hold up, *Reuters*, March 28th, 2012, online edition. http://www.reuters.com/article/2012/03/28/us-science-cancer-idUSBRE82R12P20120328

Begley, Glenn and Lee Ellis, 2012, Drug development: Raise standards for preclinical cancer research, *Nature*, 483, pp.531-533.

[4] http://catalogue.elra.info
https://www.ldc.upenn.edu/data-management/using
(consulted on November 25, 2013)

Bohannon, John, 2013, Who's Afraid of Peer Review?, *Science*, 342, pp.60-65.

Choukri, Khalid, 2013, International Standard Language Resource Number, Presentation at the *ELRA International Workshop on Sharing Language Resources: Landscape and Trends*, Paris, November 19-20.

Dickinson, Markus and W. Detmar Meurers, 2003a, Detecting Errors in Part-of-Speech Annotation. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL-03)*. Budapest, Hungary, pp. 107–114.

Dickinson, Markus and W. Detmar Meurers, 2003b, Detecting Inconsistencies in Treebanks. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*. Va¨xjo¨, Sweden.

Dickinson, Markus and W. Detmar Meurers, 2005, Towards Detecting Annotation Errors in Spoken Language Corpora, In *Proceedings of the 15th Nordic Conference of Computational Linguistics (NODALIDA 2005)*, Special Session on Treebanks for Spoken Language and Discourse.

Eskin, Eleazar, 2000, Automatic Corpus Correction with Anomaly Detection. In *Proceedings of the First Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-00)*. Seattle, Washington.

Fanelli, Daniele, 2009 How Many Scientists Fabricate and Falsify Research? A Systematic Review and Meta-Analysis of Survey Data, *PLOS ONE*, DOI: 10.1371/journal.pone.0005738.

Frezza, Bill, 2011, The Financially Driven Erosion of Scientific Integrity, *Real Clear Markets*, December 5, 3011, online edition. http://www.realclearmarkets.com/articles/2011/12/05/the_financially_driven_erosion_of_scientific_integrity_99401.html.

Hiltzik, Michael, 2013, Science has Lost its Way, at a big Cost to Humanity, *Los Angeles Times*, October 17, 2013, online edition, http://www.latimes.com/business/la-fi-hiltzik-20131027,0,1228881.column#ixzz2lT8zjZWD

Nail, Gautam, 2011, Scientists' Elusive Goal: Reproducing Study Results, *The Wall Street Journal*, December 2, 2011, online edition, http://online.wsj.com/news/articles/SB10001424052970203764804577059841672541590

Prinz, Florian, Thomas Schlange and Khusru Asadullah, 2011, Believe it or not: how much can we rely on published data on potential drug targets?, *Nature Reviews Drug Discovery* 10, 712.

Stodden, Victoria, 2013, Resolving Irreproducibility in Empirical and Computational Research, *IMS Bulletin Online*, American Institute of Mathematical Statistics. http://bulletin.imstat.org/2013/11/resolving-irreproducibility-in-empirical-and-computational-research/

Ule, Tylman and Kiril Simov, 2004, Unexpected Productions May Well be Errors. In *Proceedings of Fourth International Conference on Language Resources and Evaluation (LREC 2004)*. Lisbon, Portugal.

van Halteren, Hans, 2000, The Detection of Inconsistency in Manually Tagged Text. In Anne Abeilleé, Thorsten Brants and Hans Uszkoreit (eds.), *Proceedings of the Second Workshop on Linguistically Interpreted Corpora (LINC-00)*, Luxembourg.

Zaenen, Annie, 2006, Mark-up Barking Up the Wrong Tree, *Computational Linguistics*, 32 (4), pp. 577-580.

Zimmer, Carl, 2012, A Sharp Rise in Retractions Prompts Calls for Reform, *The New York Times*, April 16, 2012, online version, http://www.nytimes.com/2012/04/17/science/rise-in-scientific-journal-retractions-prompts-calls-for-reform.html?_r=0

Studying Interannotator Agreement in Discriminant-based Parsebanking

Helge Dyvik,[*§] Martha Thunes,[*] Petter Haugereid,[*]
Victoria Rosén,[*§] Paul Meurer,[§] Koenraad De Smedt[*]
and Gyri Smørdal Losnegaard[*]

University of Bergen[*] and Uni Research[§]
Bergen, Norway

E-mail: dyvik@uib.no

Abstract

This paper reports on a pilot study on interannotator agreement in discriminant-based parsebanking, especially with a view to uncovering linguistic issues in the grammar and lexicon. We classify the annotator discrepancies according to their causes, pointing to different stategies for avoiding them in the future, e.g. with regard to documentation or to grammar and lexicon development, and discuss a selection of examples.

1 Introduction

Many treebanking approaches combine automatic analysis tools, such as taggers and parsers, with manual editing of syntactic representations. Interannotator agreement in these approaches is usually studied by comparing labeled nodes and edges in structures produced by different annotators [1].

Parsebanking in a strict sense differs from these approaches in that annotators are only allowed to choose between automatically computed full parses but not to create, enhance or otherwise modify any structures manually. This guarantees that the parsebank is always fully in accordance with the grammar which is used to analyze the corpus [8]. In a parsebanking context, classical measures of interannotator agreement comparing labeled nodes and edges are not the best way of analyzing the annotation process.

The main problems in parsebanking are ambiguity and coverage. Sentences not covered by the grammar and lexicon are either ungrammatical or require an extension of the grammar or lexicon. Since the annotation process provides feedback for such extensions, parsebanking in our approach is also a method for incremental grammar and lexicon development, as will be discussed below [11, 9].

Ambiguity in a large coverage grammar frequently causes the number of alternative analyses to be in the hundreds, so that parse selection needs to be efficient.

Automatic analysis of the forest of alternatives produces a set of lexical, morphological and syntactic *discriminants* dividing up the set of alternatives. Annotators typically make less than ten discriminant choices to reduce the number of analyses from several hundred to one.

The automatic computation of discriminants and their selection by human annotators have been successfully used in a number of parsebanking projects [3, 7, 14, 13]. Among these, the Alpino treebank allowed manual post-editing, and interannotator agreement was studied on final, edited syntactic representations. In a parsebanking approach like ours, however, where only unedited parser outputs are stored, the treebank can be repeatedly reparsed with an updated grammar. The annotators' earlier discriminant choices will then be automatically reapplied to the new sets of analyses, and will frequently be sufficient to single out a unique analysis again.

The basis for grammar development is continual feedback from the annotators, as they encounter wrong or missing analyses of the corpus sentences. The only previous study of interannotator agreement based on discriminant selection which we know of is related to the Hinoki treebank of Japanese [13]. In that study, a quantitative and qualitative analysis of interannotator agreement was instrumental for investigating the effects of a method for speeding up annotation by using part-of-speech tags.

Our aim has been to carry out a pilot study of interannotator agreement, utilizing our new facilities for displaying annotation discrepancies in the treebank interface, primarily in order to explore the possible role of such studies in detecting and rectifying shortcomings in the grammar and lexicon, in the documentation, and in the criteria governing parse selection by the annotators. Given the limited scope of the study, the quantitative estimation of overall reliability of the annotation is a less prominent aim. The study was carried out on the INESS parsebank for Norwegian, parsed with XLE (the Xerox Linguistic Environment). The grammar is NorGram, a hand-written grammar for Norwegian based on the LFG (Lexical-Functional Grammar) formalism [2, 5]. Our study indicates that an in-depth analysis of disagreements in discriminant selection may provide a valuable basis for improving the linguistic analysis underlying the parsebanking method.

2 Studying interannotator disagreement

A parsebanking approach without post-editing guarantees consistency of the parse results with the grammar, while inconsistencies among the annotators are possible in the resolution of ambiguity. In the INESS infrastructure, the normal feedback of annotator comments reflects conscious realizations by the annotators that something in the grammar or lexicon may need adjustment. But interannotator disagreement also uncovers shortcomings that may have gone unnoticed by the annotators.

An annotator discrepancy is here understood as a difference in the selection of available parse results for a given sentence. Such differences can be identified by the discriminant choices involved in finding a unique parse result for the sentence in question. A discrepancy will arise if there is at least one such difference between

the analyses selected for that sentence. Discrepancies also occur in cases where one or more annotators have left open at least one choice made by another annotator.

In traditional treebanks, studying interannotator agreement may require special algorithms that compare the annotations in the treebank [4, 6]. We were able to use our own and XLE's existing algorithms for simultaneous display of all analyses of a sentence in the form of packed c- and f-structures (constituent structures and functional structures), which were easily modified to display differences in annotator choices. In the packed representations, common substructures are displayed only once, and each substructure is labeled (in an abbreviated form) by the set of analyses it is part of. In c-structures, these labels are realized as additional nodes, whose daughters are each labeled by the analyses they are valid for. In the annotator comparison interface, only solutions chosen by the selected annotators are displayed as packed structures. In contrast to ordinary packed solution display, substructures that are not common to all (selected) annotators are labeled by the annotators who chose them.

Figure 1 shows a minimal example of discrepancy. The analyzed sentence is (1):

(1) Hun gav gutten en dask i baken.
she gave boy.DEF a slap in behind.DEF
'She slapped the boy on his behind.'

The disagreement in Figure 1 concerns the question of PP attachment: does the prepositional phrase *i baken* 'in behind.DEF' modify the nominal *en dask* 'a slap' or the verb *gav* 'gave'? In the given example, two annotators (*gy* and *m*) have chosen the former analysis, and two (*gu* and *p*) have chosen the latter. In the c-structure the discrepancy is shown by the two alternative subtrees given for the string *gutten en dask i baken*. It can also be seen in the f-structure: the choice of verbal PP attachment selects the analysis where *i baken* is assigned the function of ADJUNCT to the three-place predicate 'gi<[], [], []>', whereas the choice of nominal PP attachment selects the analysis where *i baken* is an ADJUNCT to the predicate 'dask'. For the sake of illustration, the screenshot in Figure 1 displays only a part of the information that is available in the sentence window. [1]

3 Categories of discrepancies

Discrepancies among the annotators have various causes, pointing to different strategies for avoiding them in the future. Based on their cause we may sort the discrepancies into the following categories, among others:

1. Discrepancies arising from failure to grasp the intention behind the grammatical categorizations. The remedy in this case is better documentation and discussions among the annotators and the grammar developer in order to sharpen the criteria for making the distinction.

[1] The structures in Figure 1 are visualized using the LFG Parsebanker [12, 10].

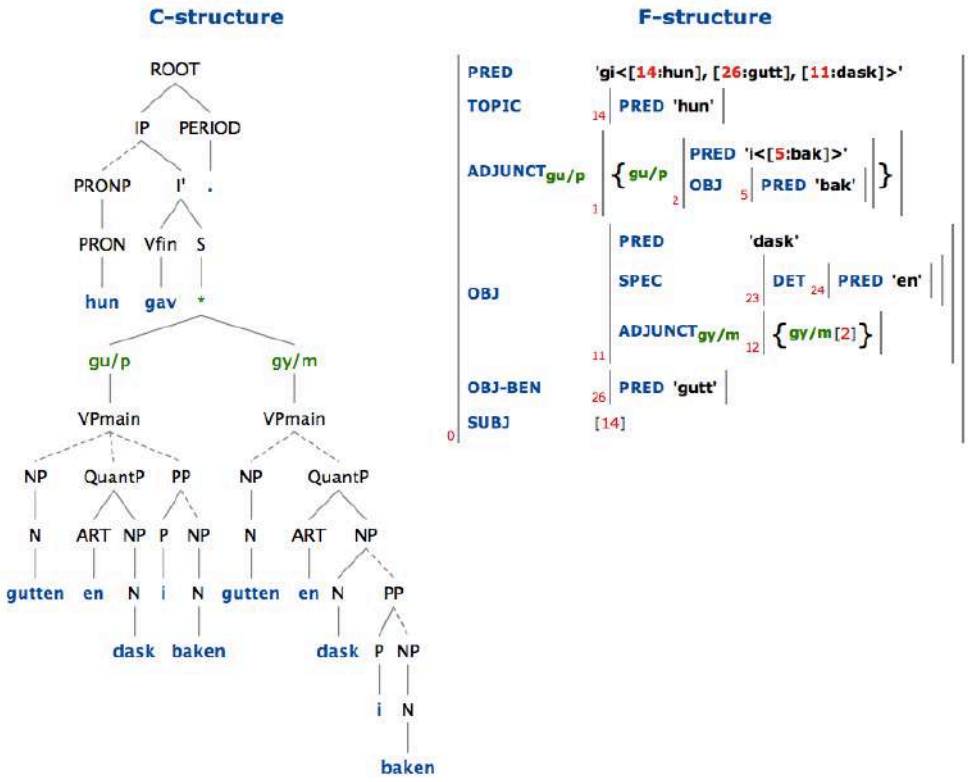


Figure 1: A screenshot showing a sentence with *one* differing discriminant choice, displayed in the c-structure as well as in the f-structure.

2. Discrepancies arising from insufficient basis in the text for making a unique decision. These cases may be assigned to one of two subcategories:

   (a) Genuine ambiguity: the distinction in question is generally well motivated, but in the case at hand there is no basis for deciding exactly what the author had in mind. In such cases the best solution may be to refrain from making a choice, ending up with more than one analysis.

   (b) Vagueness: there is no clear choice to be made between alternatives presented by the grammar and the lexicon – it is hard to see how the distinction can be applied to the case at hand, even from the author's point of view. In such cases the question arises whether it is motivated to keep the distinction in the grammar or lexicon, at least in its current form.

3. Discrepancies arising from failure of the grammar or lexicon to cover the construction under consideration, resulting in differing second-best choices among the annotators. In such cases the grammar or lexicon should probably be extended.

The 'vagueness' in 2(b) is a feature of the individual case, not necessarily of the distinction in general, which may be clear enough in other cases. However, if the number of vague cases is sufficiently high as compared to the total number of occurrences, the motivation for the distinction is weakened. Imposing conventions of default decisions is of limited value, since users of the treebank then cannot trust the annotation anyway, but will have to inspect the examples themselves. Leaving an ambiguity unresolved in a high number of instances is also unsatisfactory. Hence the suggested solution is to remove the distinction from the grammar. This will also increase the efficiency of the annotation. The linguistic aspects of such grammar modifications need to be discussed in each individual case.

4 A Pilot Study

A text of 100 consecutive sentences was selected from one of the novels represented in the treebank: *Knirk: scener fra et ekteskap* by Inge Bøgel Lassen. The sentences were disambiguated by four annotators, making discriminant-based choices among the analyses offered by the grammar.[2] The basic figures are shown in Table 1.

Although this is not a quantitative study of interannotator agreement, the number of annotated sentences being so small, it is worth pointing out that the seemingly high number of sentences with disagreement in Table 1, i.e., 44, is the number of cases where there was not complete agreement among all the four annotators. In order to make the figures more comparable to existing studies, where it is usually pairwise disagreements that are counted, we may note that the average number of sentence-level disagreements within each of the six *pairs* of annotators is 27. Partial disagreements, where at least one annotator has ended up with more than one analysis, and some but not all of these analyses are shared with the other annotator, are then also counted as full disagreements. Relating this number to the number of sentences with more than one analysis from the grammar, i.e., 68, yields an interannotator agreement score of 60.3%. If, on the other hand, partial disagreements are scored relative to the overlap of the sets of chosen analyses, the average number of sentence-level disagreements within the six annotator pairs is 25.5, which gives an interannotator agreement score of 62.1%.

Annotated sentences:	100
Sentences given a valid analysis by the grammar:	81
Among these, sentences with more than one analysis:	68
Among these, sentences with disagreement:	44

Table 1: Classification of sentences according to number of analyses and annotator disagreement.

In the 44 sentences with annotator discrepancies, a total of 61 differing discriminant choices were involved: 31 sentences differed in only one discriminant

[2] We are indebted to Gunn Inger Lyse, University of Bergen, who joined us in preparing the data for this investigation.

choice, 9 sentences in two, and 4 sentences in three. A classification of the differing discriminant choices is shown in Table 2, referring to the linguistic phenomena involved in the choices.

Category	**Freq**
det, den, de as expletive pronoun, referring pronoun, article or demonstrative	15
Different adjunct attachments	12
Different syntactic functions	5
Different lexical categories	5
Lexical, grammatical or selected preposition	4
noe 'some' as mass or countable	3
Multi-word expression or compositional phrase	3
Different lemma assignments	3
Different morphological categories	2
Coordination or quasi-coordination	2
Inquit or non-inquit verb	1
Different coordination levels	1
Grammar ambiguities without linguistic motivation	5
Total	61

Table 2: Classification of the 61 differing discriminant choices.

The most frequent disagreement type in Table 2 concerns the forms *det* 'it'/'that'/'the', *den* 'it'/'that'/'the', and *de* 'they'/'those'/'the' and is thus probably fairly language-specific, while the second most frequent type, adjunct attachment, presumably exemplifies a more universal problem. The first type involves choices among personal and impersonal constructions and between pronoun and demonstrative readings, and hence often involves rather delicate judgments. The second type involves both PP, adverb and adverbial clause attachments. We will consider some examples and relate them to the types of discrepancies specified in Section 3 in order to determine their relevance for further grammar development.

The last category in Table 2, *Grammar ambiguities without linguistic motivation*, is of limited interest here since it does not reflect linguistic ambiguities, but rather ambiguities arising from technical aspects of the formal grammar. Some cases concern redundancies that should be removed, e.g., the (linguistically) same analysis being inadvertently covered in two different ways by the grammar. Other cases concern redundancies which are more difficult to get rid of, e.g. relating to punctuation, which may be allowed in the same position by different rules and therefore sometimes may leave the annotators vacuous choices.

4.1 Discrepancies of type 1: the criteria need sharpening

In cases where it is difficult to grasp the intention behind grammatical categorizations because of insufficiently developed criteria of selection, annotators may choose different analyses for the same sentence. This may be illustrated by cases

where the challenge for the annotators is to decide whether instances of the third person singular pronoun *det* 'it' is a referring expression or a non-referential expletive when occurring together with the verb *være* 'be'. These may be copula constructions, presentative sentences, or various types of impersonal constructions. The issue can be demonstrated by examples (2) and (3):

(2) Det var noe romantisk tøv.
it was some romantic nonsense
'It was some romantic nonsense.'

(3) Det var helg.
it was weekend
'It was the weekend.'

In cases like this, tests may be applied to find the correct analysis: is the word *det* anchored to an identifiable discourse referent in the preceding context, and does it express an argument in a predicate-argument structure? If yes, it is a referring pronoun (PRON). If not, it is an expletive pronoun (PRONexpl). The next question is whether the existential reading of the verb *være* is suitable in the given context, in which case it is contextually appropriate to replace *det* with the existential expletive *der* 'there', and *være* with the synonymous verb *finnes* 'exist'. This normally identifies the presentative construction. If such replacement results in an unidiomatic sentence, or clearly changes the meaning, then we are faced with an impersonal construction involving a zero-valued predicate, like English *It's raining*, *It's cold*.

For examples (2) and (3) these tests have fairly clear answers. Still, some of them were not applied by one or more annotator, because in these sentences our experiment showed discrepancies with regard to the choice between PRON and PRONexpl. In (2) one annotator chose the expletive reading of *det*, while the other three selected the referential reading. Inspecting the context reveals that in (2) *det* refers to the content of a film that has been talked about. The sentence is a copula construction, and the referent of *det* is the entity of which 'being some romantic nonsense' is predicated. Concerning (3), one annotator picked the referential reading of *det*, and three chose the expletive. As this sentence is the opening of a new scene in the narrative, there is no plausible referent for *det*. If *det* is a non-referential expression, it is an expletive pronoun, and does not have argument status. Therefore (3) cannot be a copula sentence. A presentative analysis can also be ruled out, since the existential reading of *være* appears odd in this case: the sentence *Der fantes helg* ('There existed a weekend') is not idiomatic. The conclusion is that (3) is an impersonal construction, with only one argument in the predicate-argument structure, expressed by *helg*. [3]

There are, however, some cases where it may be more difficult to decide on the correct analysis. In (4) the contextually appropriate reading is a copula construction

[3]The predicate-argument structures in LFG closely correspond to the verbs of the language, usually assigning a predicate to every verb, even the copula. In a semantic representation the predicative complement *helg* would not be an argument, but rather the predicate itself, with 0 arguments, like 'rain' in *It's raining*.

where *det* is a referring pronoun, and the property of being no problem is predicated of the referent of *det*.

(4) – Det er vel ikke så farlig.
– it is well not so dangerous
'That's not really a problem, is it?'

In contrast to sentence (2), it is somewhat less clear in (4) how to identify the referent of *det*. The pronoun is not anchored to a specific entity, but to a certain state of affairs which is implicit in the discussion that is reported in the narrative. If a discourse referent is not readily identifiable for *det* in (4), an impersonal reading may appear plausible, and such a reading was in fact chosen for this sentence by one of the four annotators.

Further, in sentence (5) the impersonal construction is the contextually appropriate reading:

(5) Nå var det virkelig for mye.
Now was it really too much
'Now it was really too much.'

In the case of (5), two annotators chose the referential reading of *det*, and the other two picked the intended, expletive reading. By considering the textual context immediately preceding (5), the referential choice becomes understandable. The text describes a stressful situation, and this very situation can possibly be perceived as the referent of *det*. Also, it is not easy to find clear criteria to distinguish between a case like (4), where *det* does refer to a described situation or state of affairs, and the case of (5), where it does not.

In order to amend annotator discrepancies of the kind illustrated here, it seems necessary to sharpen the criteria for distinguishing between the different constructions. Another test could be introduced to help in identifying impersonal constructions. This applies to cases where the existential reading can be ruled out, and where it is difficult to decide whether or not *det* refers to an entity or some state of affairs in the preceding discourse. In such instances another test could be to try to replace *det* with a demonstrative. If that changes the meaning, or yields a contextually inappropriate sentence, it indicates that *det* is non-referential, supporting the impersonal reading. For example, in (5) this could be tested by pronouncing *det* with emphatic stress, which brings about the demonstrative reading of *det* 'that'. Given the context, this appears odd. This strengthens the position that there is no referent of *det* in the preceding discourse, and indicates that (5) is an impersonal construction.

In the present study we need not go more deeply into the various types of Norwegian *det er* constructions. The relevant aspect here is that sharpened criteria may improve annotators' intuitions, and hence reduce discrepancies in further practice.

4.2 Discrepancies of type 2(a): the choice should be left open

Example (6) illustrates the type of disagreement where the choice should be left open.

(6) Hun følte ikke noe.
she felt not anything
'She didn't feel anything.'/'She felt nothing.'

In (6) the annotators were given the choice between two alternative attachments for the negative adjunct *ikke*, which can be taken to modify either the verb *følte* or the quantifier *noe*. Two annotators opted for modification of the verb *følte*, while the other two chose modification of the quantifier *noe*. If *ikke* is analyzed as modifying the quantifier, the word *noe* would be stressed. If *ikke* on the other hand is analyzed as modifying the verb, then *noe* would most likely be unstressed. There is a perceptible (but slight) semantic difference between the alternatives, but both alternatives seem possible in this context. Without access to prosody the intended analysis is hard to determine. The conclusion is to include in our criteria that in such cases the choice should be left open and the result marked as 'gold', which means that both remaining analyses will be treated as valid.

4.3 Discrepancies of type 2(b): the grammar should remove or modify the distinction

In example (7) a distinction is revealed that could be removed from the grammar.

(7) Hun kjente ham selvsagt også uten briller.
she knew him of course also without glasses
'She of course also knew him without glasses.'

In (7) the annotators are presented with the choice between two attachments for the sentence adverb *også* 'also'. Three of the four annotators picked the analysis where *også* is a daughter of S and hence an adjunct of the verb *kjente*. The fourth annotator selected the analysis where *også* is a daughter of the PP *uten briller* and hence an adjunct of the preposition *uten*.

In this case it is hard to grasp a clear semantic distinction between the alternatives. Under both of them the focussed PP falls within the scope of the adverb. The reason why the annotators are presented with this second option, where the adverb is allowed as a daughter of PP, is that constituents like *også uten briller* may appear in sentence initial position, where there is no alternative but to analyze *også* as a daughter of the PP, and hence the grammar must allow such PPs. Examples like this raise the question whether this kind of constituent should be limited to certain positions, such as sentence initial position and right dislocation. A limitation of this kind can then be introduced tentatively, pending further corpus experience.

4.4 Discrepancies of type 3: The grammar needs extending

An example revealing the need to extend grammar coverage is sentence (8).

(8) – Er det slik du vil ha det?
– is it thus you want have it?
'Is this the way you want it?'

Again, the annotators were offered the choice between the referential PRON reading and the expletive, non-referential PRONexpl reading of the first *det*. The choice of referential PRON gives the reading 'Is it (i.e., the thing we're talking about) the way you want it?'. One annotator chose this reading. The choice of expletive PRONexpl gives the reading as an extraposition, as in an English sentence like *Is it good that you want it?*, which is almost hard to grasp intuitively when *good* is replaced with *thus*, as it is here; the meaning is the awkward 'is the fact that you want it thus?'. Three annotators chose this reading.

However, neither of the readings offered is adequate. The context makes it clear that there is no referent available motivating the referential PRON reading. Still, the awkward extraposition reading is not the one we want, either. The reason why it was still chosen by some annotators was obviously the correct observation that the subject was expletive, while the adequate expletive construction had not been found. As indicated by the translation of example 8, this is a cleft sentence: 'is it thus you want it?', i.e., 'is this the way you want it?' The grammar did not cover this kind of cleft sentence, in which an adjective is the focussed element. The annotator disagreement brought attention to this shortcoming, which could hence be rectified.

Another example is (9):

(9) – Det var ikke noe.
– it was not something
'It was nothing.'

In this case three annotators selected a referential reading of the subject, whereas one chose the existential reading. A referential reading requires a referent of *det* in the context, but there is no clear candidate in the preceding discourse, apart from possibly the described situation itself. Testing for an existential reading involves replacing *det* with *der* and *var* with *fantes*. This gives the sentence *Der fantes ikke noe* 'There was nothing', which is perfectly idiomatic, but contextually inappropriate. The intended reading of (9) is an impersonal construction, where *det* is an expletive pronoun, and the quantifier *noe* expresses the only argument in the predicate-argument structure. In this case this reading was, however, not available among the analyses delivered by the parser. The grammar had to be extended to cover the possibility of nominal predicative complements to impersonal *være*.

5 Conclusions and outlook

Our in-depth qualitative pilot study of interannotator agreement has provided us with information pertinent to the distinctions underlying the LFG grammar for Norwegian which is being incrementally developed together with the parsebank.

Table 3 shows the result of sorting the discrepancies found in our study according to the categories presented in Section 3. In addition to the types 1, 2(a)-(b), and 3 which we have discussed and illustrated, Table 3 includes two categories of a rather residual kind. Firstly, there are some instances of discrepancies where the mistaken choices are obvious performance errors, such as picking the syntactic category YEAR, instead of NUMdig for the numerical expression *20.45*. These are

counted as annotator errors in Table 3. Secondly, some discrepancies have arisen because of vacuous choices as explained in Section 4, or due to technical shortcomings. These instances are listed at the bottom of Table 3 as 'linguistically vacuous choices'.

Type 1: the criteria need sharpening	35
Type 2(a): the choice should be left open	3
Type 2(b): the grammar should remove or modify a distinction	8
Type 3: the grammar needs extending	4
Annotator errors	6
Linguistically vacuous choices	5
Total	61

Table 3: Number of differing discriminant choices in each category (Section 3).

The figures in Table 3 suggest that most cases of annotator disagreement are rooted in insufficient documentation of the criteria for the distinctions made in the grammar. But they also indicate that interannotator agreement studies are a promising source of information about grammar and lexicon shortcomings which have gone unnoticed by the annotators during the annotation process. Therefore repeated studies of this kind should be included in the work cycle of the project. Future work will include a more extensive quantitative study on discriminant-based interannotator agreement.

References

[1] Thorsten Brants. Inter-annotator agreement for a German newspaper corpus. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece, 2000.

[2] Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. The Parallel Grammar project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation, Taipei, Taiwan*, 2002.

[3] David Carter. The TreeBanker: A tool for supervised training of parsed corpora. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 598–603, Providence, Rhode Island, 1997.

[4] Markus Dickinson and W. Detmar Meurers. Detecting inconsistencies in treebanks. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*, The Ohio Sate University Department of Linguistics, 2003.

[5] Helge Dyvik. Nødvendige noder i norsk: Grunntrekk i en leksikalsk-funksjonell beskrivelse av norsk syntaks [Necessary nodes in Norwegian: Basic properties of a lexical-functional description of Norwegian syntax]. In

Øivin Andersen, Kjersti Fløttum, and Torodd Kinn, editors, *Menneske, språk og felleskap*. Novus forlag, 2000.

[6] Seth Kulick, Ann Bies, Justin Mott, and Mohamed Maamouri. Using derivation trees for informative treebank inter-annotator agreement evaluation. In *Proceedings of NAACL-HLT 2013*, pages 550–555, Atlanta, Georgia, June 9–14 2013. Association for Computational Linguistics.

[7] Stephan Oepen, Dan Flickinger, Kristina Toutanova, and Christopher D. Manning. LinGO Redwoods, a rich and dynamic treebank for HPSG. In Joakim Nivre and Erhard Hinrichs, editors, *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories*, pages 117–128. Växjö University Press, 2003.

[8] Victoria Rosén and Koenraad De Smedt. Theoretically motivated treebank coverage. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NoDaLiDa-2007)*, pages 152–159. Tartu University Library, Tartu, 2007.

[9] Victoria Rosén, Koenraad De Smedt, and Paul Meurer. Towards a toolkit linking treebanking to grammar development. In *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories*, pages 55–66, 2006.

[10] Victoria Rosén, Koenraad De Smedt, Paul Meurer, and Helge Dyvik. An open infrastructure for advanced treebanking. In Jan Hajič, Koenraad De Smedt, Marko Tadić, and António Branco, editors, *META-RESEARCH Workshop on Advanced Treebanking at LREC2012*, pages 22–29, Istanbul, Turkey, May 2012.

[11] Victoria Rosén, Paul Meurer, and Koenraad De Smedt. Constructing a parsed corpus with a large LFG grammar. In *Proceedings of LFG'05*, pages 371–387. CSLI Publications, 2005.

[12] Victoria Rosén, Paul Meurer, and Koenraad De Smedt. LFG Parsebanker: A toolkit for building and searching a treebank as a parsed corpus. In Frank Van Eynde, Anette Frank, Gertjan van Noord, and Koenraad De Smedt, editors, *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT7)*, pages 127–133, Utrecht, 2009. LOT.

[13] Takaaki Tanaka, Francis Bond, Stephan Oepen, and Sanae Fujita. High precision treebanking—blazing useful trees using POS information. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 330–337, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[14] Leonoor van der Beek, Gosse Bouma, Robert Malouf, and Gertjan van Noord. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN) 2001*, Twente University, 2002.

Sampling Methods in Active Learning for Treebanking

Masood Ghayoomi[†] [‡] Jonas Kuhn[‡]
[†]Department of Mathematics and Computer Science, Freie Universität Berlin
[‡] Institute for Natural Language Processing, University of Stuttgart

Masood.Ghayoomi@fu-berlin.de Jonas.Kuhn@ims.uni-stuttgart.de

Abstract

In this paper, we propose using active learning for treebanking of the Persian language to find informative samples to enlarge the treebank. We propose a new uncertainty sampling method, called combo uncertainty model, which works based on word- and class-based parsing models and a meta-ranker on the top of the two models. In this paper, we show that the combo uncertainty model outperforms the individual word- and class-based models. Additionally, we use these two parsing models for a query-by-committee sampling method. Experimental results show that the combo uncertainty sampling method is the best approach when developing a treebank with minimum number of sentences, but the query-by-committee method is a better choice when the number of learnt words is taken into consideration.

1 Introduction

Availability of annotated data plays a very important role in supervised methods to process a language automatically. Developing such data set is a very time consuming and tedious task that requires a high amount of a human effort to annotate the data. Active Learning (AL) is a supervised, machine learning method used for selecting hard samples from a data pool, and it asks an oracle to annotate this portion of data [20]. This learning method is different from passive learning in which large amount of data should be annotated without using any intelligence to select data. AL is widely used in various applications of Natural Language Processing (NLP) that require annotated data, such as parsing [12, 1], information extraction [25], semantic role labeling [4], machine translation [8], and name entity recognition [14] to be named some.

A statistical parser requires a treebank to learn the grammar of the corresponding language and apply the patterns at the unseen data. Languages like English and German are rich in terms of the availability of large treebanks to train parsers.

But there are languages like Persian which suffer from lack or small size of such data source. AL is a great help to enrich the resources of the less considered languages. To reach the goal, informative samples which are hard for parsers should be annotated by an oracle and be added to the training data, i.e. a treebank. In this paper, several sampling models within uncertainty and query-by-committee sampling methods are employed to find the informative samples. The selected samples are annotated by an oracle and added to the treebank which is the training data of a parser. The enlarged treebank improves the parsing performance consequently.

The structure of this paper is as follows: Section 2 describes the sketch of AL along with the learning scenarios and the sampling methods. Section 3 explains our AL models. The experimental setup and the obtained results are reported and discussed in Section 4. Background on using AL for parsing and treebanking is described in Section 5. The paper is finally summarized in Section 6.

2 Active Learning

2.1 Learning Scenarios

Stream-based [5] and pool-based [15] models are the two major learning scenarios in AL which are very popular among researchers and frequently used in various NLP applications. We use the pool-based sampling scenario for our application such that in each iteration the learner first takes all the samples from the data pool, and then it ranks them descendingly based on a selection criterion. Finally, it selects top k samples from this list and hands the selected samples out to an oracle for annotation.

2.2 Sampling Methods

Settles [20] introduced a number of sampling methods in AL. Among them, uncertainty [15] and query-by-committee [21] sampling methods are the most popular ones which are widely used for NLP applications. In uncertainty sampling, only one learner is employed and the samples that the learner has the least confidence on their annotation are selected according to a criterion and handed out to an oracle. Entropy [22] is the most popular criterion to select samples for NLP applications, such as parsing [1, 12]. In the query-by-committee sampling method, more than one learner is used such that each learner proposes an annotation for each unannotated sample. Then, among the samples, those which have the highest degree of disagreement on the annotations among the committee of learners are selected as informative samples, and they are handed out to an oracle for annotation.

3 Our Proposed Sampling Methods

3.1 Uncertainty Sampling

Algorithm 1 displays the steps for selecting uncertain samples in our models. In our AL application for treebanking, the learner is a parser that learns the grammar of the Persian language.

Algorithm 1 Active Learning with Entropy-based Uncertainty sampling

Input: Seed data S from the Persian HPSG-based treebank, Pool of unlabeled samples U
repeat
Step1: Use S to train the parser P
Step2: Use P to parse U, and extract *n*-best (*n*=20) parse trees for each sentence
Step3: Compute tree entropy TE of each sentence based on the probability score of the *n*-best parse trees
Step4: Sort descendingly the sentences based on their TE score
Step5: Select top K samples from the sorted parsed trees
Step6: Augment S with K samples, and remove K from U
until the stopping criterion is met

We use the entropy-based sampling method introduced by Hwa [12] to select uncertain samples for enlarging the treebank. In each iteration of this method, the entropy of each sentence is calculated based on the probability scores of its *n*-best (*n*=20) candidate parse trees. If a grammar is certain about the structure of a sentence, then one parse tree will be assigned a high score and the rest a low score which results in a low entropy. While for uncertain sentences, all possible parse trees will have relatively uniform scores and a high entropy consequently. The Tree Entropy (TE) of sentence s is calculated according to Equation 1.

$$\begin{aligned} TE(s,G) &= -\sum_{v \in V} p(v) log_2(p(v)) \\ &= -\sum_{v \in V} \frac{P(v|G)}{P(s|G)} log_2(\frac{P(v|G)}{P(s|G)}) \end{aligned} \tag{1}$$

where V ($v \in V$) is the set of possible parses of s. $P(v|G)$ is the probability score of parse tree v which is provided by the parser. $P(s|G)$ is the sum of probabilities of its parses:

$$P(s|G) = \sum_{v \in V} P(v|G) \tag{2}$$

In our study, uncertainty sampling methods use Algorithm 1. In the first two models, word- and class-based parsing described below, solely use the tree entropy score as a criterion for selecting uncertain samples. In the third model, we employ the two models and add a step to Algorithm 1 to use a meta-ranker on top of the word- and class-based products to select the uncertain samples.

Model 1: Word-Uncertainty In this model, we use a normal word-based parsing model; i.e., the actual sentences in the treebank are used for training the parser, and top k samples with high entropy are selected. This model is also evaluated based on the word-based parsing.

Since the word-based parsing suffers from the data sparsity problem at both lexical and syntactic rule levels, the k selected samples contain informative lexicon and syntactic rules for the parser. We propose a model to reduce this complexity.

Model 2: Class-Uncertainty In Model 1, two levels of data sparsity, namely the lexical and syntactic rule sparsity, are taken into consideration simultaneously, and sentences which are more sparse are selected first. To reduce the effect of lexical sparsity and put a higher priority on sentences with informative syntactic rules, we use class-based parsing.

In class-based parsing, we first use a word clustering algorithm, such as the Brown word clustering [3], to cluster the lexical items of a corpus in an off-line mode. Since the number of clusters are much less than the number of words, the class-based model provides a more coarse-grained representation of the lexical knowledge which results in less data sparsity at the lexical level. Then, all words in the sentences of the treebank are mapped to their corresponding clusters. Afterwards, the mapped sentences are used for training a parser and parsing the unannotated samples which are also converted to the corresponding clusters.

As shown in previous studies including Ghayoomi [7], the class-based parsing improves the parsing performance compared to the normal word-based parsing. Ghayoomi [7] pointed out a shortcoming of the Brown word clustering [3]. In this algorithm, homographs are treated equally. To resolve this problem, Ghayoomi [7] added the part-of-speech tags to the words to distinct homographs, and then performed the clustering approach. After mapping the words of the treebank into their corresponding clusters, Algorithm 1 is used for selecting k informative sentences.

To make the results comparable with other models and have a fair comparison, the product of the class-based parsing is reconverted to the word-based format at the end, and the model is finally evaluated based on the word-based parsing. The advantage of the class-based parsing is reducing the lexical sparsity and minimizing its impact on sampling.

Model 3: Combo-Uncertainty Although uncertainty sampling normally performs based on a single parser or parsing model, we propose a new uncertainty sampling model which works based on the two models, called combo-uncertainty. This model is a combination of Models 1 and 2. In this model, after parsing sentences, computing their entropy, and ranking them descendingly in both models separately, we use a meta-ranker on top of the two models to re-rank sentences based on their average rank. Therefore, instead of the entropy score, each sentence is assigned a score which is the average rank of the sentence in the two models.

As an example, assume that Sentence S is parsed with Models 1 and 2. Their

outputs, S_w and S_c, obtain the ranks 12 and 4 in the descendingly sorted outputs. Using the meta-ranker on the top of the models, the score 8 is assigned to S and the position of S will be re-ranked based on the assigned score. After re-ranking sentences, top k sentences from the re-ranked list are selected as the most informative samples. The advantage of this model is that the properties of Models 1 and 2 are taken into consideration for selecting samples.

3.2 Query-by-Committee Sampling

Model 4: Combo-Committee In this model, a combination of word- and class-based parsing is employed in such a way that sentences are first parsed in both models separately without using any criteria for sampling. Then, the output of the class-based model is converted into its equivalent word-based model. This conversion makes the products of the two parsers comparable.

To find the disagreement between the learners, we measure the disagreement degree similar to the $F-complement$ metric proposed by Ngai and Yarowsky [17]. The main idea of the $F-complement$ metric is comparing the output of two learners while assuming one of them as gold data and the other one as guess data. This comparison makes it possible to calculate the F-measure between the two models for each pair of parses of a sentence, and the complement of this value is considered as the disagreement score. To select the disagreed candidates in our model, we do not compute this disagreement score, but the output of the word-based parsing as gold data is compared with the output of the class-based parsing as guess data, and their F-measure is calculated. In the next step, the sentences are sorted ascendingly based on their obtained F-measure. Then, the top k sentences which obtain the lowest F-measure are selected as the most informative sentences for which the two parsing models disagreed with the provided analyses and the parsers need such data to be trained with.

3.3 Baselines

In addition to our proposed models, we use two sampling methods as baselines: random sampling and sentence length. In random sampling, k sentences are randomly selected in each iteration, and they are given to an oracle for annotation. In sentence length sampling, sentences in the data pool are first ranked decendingly based on the number of involved words, and then the longest k sentences are selected as informative samples to be annotated by the oracle.

4 Results and Discussion

4.1 Experimental Setup

In all of our AL experiments, we use the Stanford constituent parser [13] as a learner. The HPSG-based Persian Treebank [6] which contains 1028 trees is used

for training and testing the parser. The data format of this treebank is XML which is converted into its equvalent Penn style format [7]. Gold part-of-speech tags are used in the experiments to avoid the negative interfering of tagging on parsing. The data of the Bijankhan Corpus[1] and the Persian Linguistic DataBase[2] are used for word clustering utilizing the SRILM toolkit [24] which contains the implementation of the Brown word clustering algorithm [3]. Experimentally we found out that the class-based parsing of the extended clustering model described in Ghayoomi [7] obtains the highest performance of the parser with 1000 clusters. This number of clusters is set to build the class-based model.

The reported results are based on the simulation of AL to be able to select the best approach for a real application to enlarge the existing Persian treebank. To this end, we divide the current treebank into three data sets such that 20% of the data is considered as the seed data for initialization, 10% as the test data for iterative evaluation and drawing the learning curve, and the rest is assumed as an unannotated data in the data pool to be annotated. Since our models are processed iteratively, k informative sentences (k=10) are selected in each iteration. 10-fold cross validation is used for evaluating our models, and the results of the models are compared with the baselines.

4.2 Results and Discussion

Figure 1 represents the learning curves of the entropy-based sampling models (Models 1 to 3), the query-by-committee model and the random sampling as the baseline. To make the results more readable, instead of representing the result of each iteration, the average performance for each 5 iterations is shown in this figure.

As can be seen in Figure 1, the four proposed models have beaten random sampling. This result indicates the superiority of using AL models towards our goal which is increasing the size of the training data with informative samples. In the early iterations, the results of Models 1 and 2 are compatible, but Model 3 outperforms the two models. In iteration 31, Model 1 beats Model 2, and later in iteration 35, it beats Model 3, and constantly this model has a better performance in the second half of the annotation process. Model 4 has a slightly better performance than the baseline in early iterations. Although it loses against the baseline in iteration 5, it beats all of the models in iteration 40 and the performance of the parser improves as the treebank enlarges.

Since it is expected to find the informative samples in early iterations of the AL process, the first half of the annotation process is more important than the second half. Comparing Models 1 to 4 and the baseline, we can conclude that Model 3 that has a meta-ranker can be chosen as the best method for selecting informative sentences to enlarge the treebank and create a more accurate grammar model.

A drawback of the entropy-based sampling is that sentences which are relatively long and might contain complex constructions are selected as informative

[1] http://ece.ut.ac.ir/dbrg/bijankhan/

[2] http://pldb.ihcs.ac.ir/

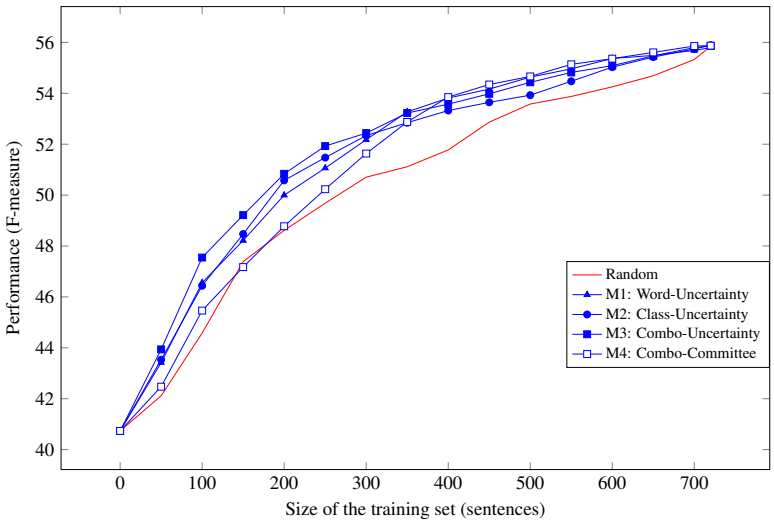


Figure 1: Learning curves of the learnt sentences in uncertainty sampling models and random sampling baseline

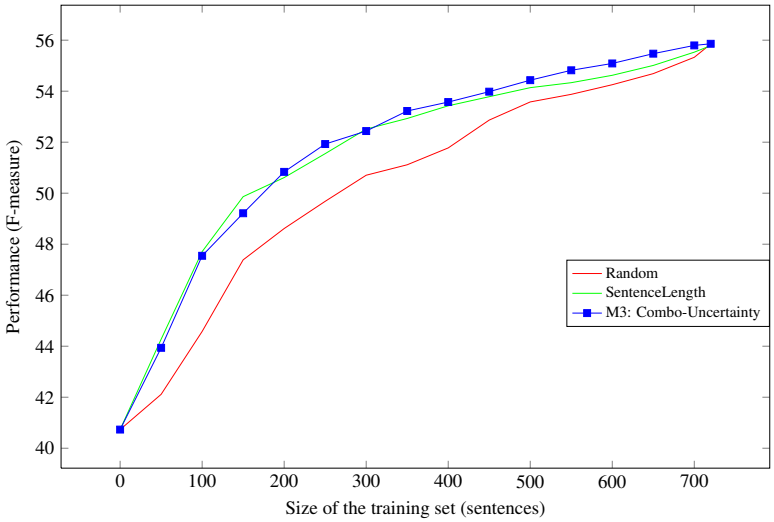


Figure 2: Learning curves of the learnt sentences in Combo-Uncertainty model and baselines

samples. To study this point, we use sentence length as the second baseline and compare Model 3 with it. As can be seen in Figure 2, the sentence length model has a relatively comparable performance with Model 3 which indicates that the performance of entropy-based sampling is compatible with a naive model that uses no intelligence to select informative sentences but only their length.

To prove this idea that the length of sentences are effective in selections of entropy-based sampling, we display in Figure 3 the learning curves of Models 3 and 4 and the baselines based on the learnt words rather than the sentences (see the x-axis).

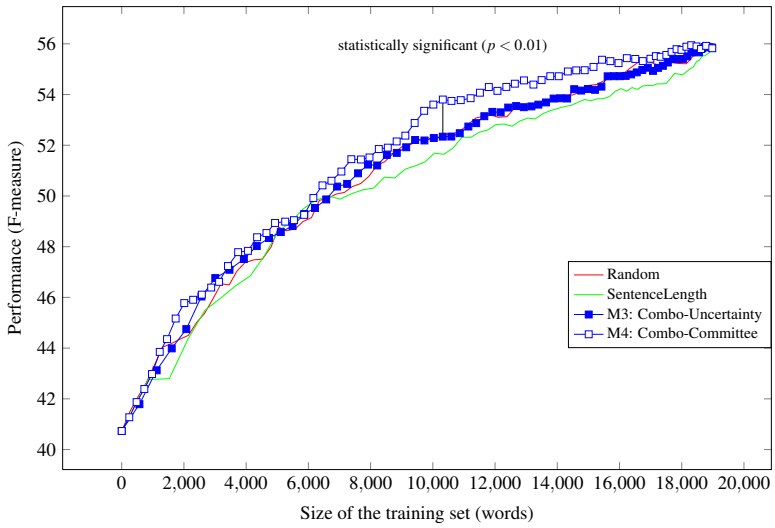


Figure 3: Learning curves of the learnt words in Combo-Uncertainty and Combo-Committee models and baselines

As shown in this figure, there is a competition between the models in the early iterations. Model 3 has a compatible performance with random sampling, and the sentence length baseline has the worst performance which indicates that learning long sentences does not guarantee to be informative for the treebank and the parser consequently. Model 4 outperforms Model 3 and the baselines in the iteration 23 by learning minimum 6400 words which are almost third of the words in the data pool. This model outperforms Model 3 and the baselines in the iteration 36 where almost half of the words in the data pool are learnt. The difference between the performance of the parser in this iteration in Model 4 and the performance of the Model 3 and the baselines with the equal number of learnt words is statistically highly significant according to the two-tailed t-test ($p < 0.01$). In this model, a higher performance is obtained by learning a minimum number of words from the data pool. This curve indicates that Model 4 is the best model for treebanking with respect to the number of words to be learnt. This also indicates that the selected samples from the data pool and adding their annotation to the training data are effective in the performance of the parser to create a more accurate grammar model and obtain a higher performance with a minimum number of words.

5 Previous Studies on AL for Parsing and Treebanking

AL is widely used for various NLP applications including parsing, either constituency or dependency, to reduce a human effort in data annotation. In the followings, we express briefly the sketch of the previous studies on using AL for constituency parsing and treebanking.

Ratnaparkhi [19] proposed an AL model which uses a maximum entropy parser. This parser selects the candidate parse trees with maximum entropy probability

from a set of derived parsed trees. In the maximum entropy framework, features are required as evidences to build the model. These features provide contextual information represented as chunks. The less specific contextual information is more interesting in this AL model to "provide reliable probability estimates when the words in the history are rare" [19]. The advantages of this sampling method are reducing the impact of sentence length on the selection and building an independent domain model.

Hwa [10, 11, 12] proposed a sampling method for inducing probabilistic lexicalized tree insertion grammars. In each iteration of the model, the entropy of each sentence is calculated based on the probability scores of its candidate parse trees. See Section 3.1 for more detail of this study.

Steedman et al. [23] proposed a co-training model in such a way that two different parsers are employed for sampling without any manual annotation. In each iteration of the model, a small set of sentences are pulled out from the data pool and stored in a cache. Then, the parsers parse the sentences in the cache. After that, a subset of the parsed sentences is selected and added to the training data. The selected data is the product of one parser which is added to the training data of the other parser. During the selection step, one parser first acts as a teacher and the other as a student, and then the roles are reversed.
The most important issue in this model is that the selection process is based on the accuracy score rates. To this end, two scoring functions are defined: (a) a scoring function based on the F-measure of the analysis against the gold data, and (b) a scoring function based on the conditional probability of the parser. Three sampling methods are exploited in the model: (a) defining a score as a threshold to select the most accurate analyses, (b) computing the difference score of the teacher and the student to choose the candidates in which the teacher is more accurate, and (c) finding the intersection between the n percent highest scores of the teacher and the n percent lowest scores of the student for the same sentence to select sentences which are accurately parsed by the teacher and incorrectly by the student.

Baldridge and Osborne [1, 2], and Osborne and Baldridge [18] described a method for AL of HPSG parse selection. They used the sentence entropy-based uncertainty sampling, a query-by-committee sampling method between log-linear and perceptron algorithms as the learners, and a combined selection method that takes the intersection of entropy- and disagreement-based models. Features are required to build the models. Two feature selection methods are exploited: (a) selecting features from derivation trees, and (b) extracting n-gram features from flattened derivation trees which are treated as a sequence of rule names.

Hughes et al. [9] described a system for selecting samples based on the combinatory categorial grammar through an interactive correction process. In this model, human annotators have an interaction with the system such that they add the constraints to the parser that return the most probable parse results which have satisfied all constraints. The informative samples are selected via a pool-based AL process in a query-by-committee model.

Lynn et al. [16] built a query-by-committee model for developing the Irish

dependency treebank. To this end, the disagreement between a committee of two parsers, namely the Malt and Mate dependency parsers, was considered as the sampling criterion.

6 Summary

In this paper, we employed the two well-known sampling methods, namely uncertainty sampling and query-by-committee, in AL for treebanking of Persian. To this end, we used word- and class-based parsing and the combination of both. Additionally, sentence length and random sampling are used as the baselines.

For uncertainty sampling, we used three entropy-based sampling methods in such a way that the word- and class-based parsing and the combination of the two with a meta-ranker on top were employed to select the informative samples from the data pool which are informative for the treebank and the parser as a result. The results showed that the entropy-based sampling method with a meta-ranker outperformed the other two models and the baselines.

Although entropy-based sampling methods outperformed random sampling, comparing its result with sentence length baseline determined that selecting relatively long sentences in entropy-based models is the main reason to achieve a better performance. To tackle this shortcoming, we proposed a query-by-committee sampling method. In this model, the two parsing models, word- and class-based parsing, were used for finding the informative samples without the impact of sentence length on selection. In this model, the sentences that the parsers disagreed with their analysis and obtained a low F-measure are selected and added to the treebank. This model selected the informative samples disregarding the sentence length, and increased the performance of the AL process with minimum growth of the words to be learnt. This model can be used for further data annotation to increase the size of the current Persian treebank with minimum words.

7 Ackowledgement

In this research, Masood Ghayoomi is funded by the German Research Foundation (DFG) via the SFB 732 “Incremental Specification in Context”.

References

[1] Jason Baldridge and Miles Osborne. Active learning for HPSG parse selection. In *Proceedings of the 7th Conference on Natural Language Learning at Human Language Technology-Conference of the North American Chapter of the Association for Computational Linguistics*, pages 17–24, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[2] Jason Baldridge and Miles Osborne. Active learning and logarithmic opinion pools for HPSG parse selection. *Natural Language Engineering*, 14(2):191–222, 2008.

[3] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479, 1992.

[4] Bertjan Busser and Roser Morante. Designing an active learning based system for corpus annotation. In *Revista de Procesamiento del Lenguaje Natural*, number 35, pages 375–381, 2005.

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

[6] Masood Ghayoomi. Bootstrapping the development of an HPSG-based treebank for Persian. *Linguistic Issues in Language Technology*, 7(1), 2012.

[7] Masood Ghayoomi. Word clustering for Persian statistical parsing. In Hitoshi Isahara and Kyoko Kanzaki, editors, *Advances in Natural Language Processing*, volume 7614 of *Lecture Notes in Computer Science: JapTAL '12: Proceedings of the 8th International Conference on Advances in Natural Language Processing*, pages 126–137. Springer Berlin Heidelberg, 2012.

[8] Gholamreza Haffari and Anoop Sarkar. Active learning for multilingual statistical machine translation. In *Proceedings of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*, Singapore, 2009.

[9] Baden Hughes, James Haggerty, Saritha Manickam, Joel Nothman, and James R. Curran. A distributed architecture for interactive parse annotation. In *In Proceedings of the Australasian Language Technology Workshop*, pages 207–214, 2005.

[10] Rebecca Hwa. Sample selection for statistical grammar induction. In *Proceedings of the 2000 Joint special interest group for linguistic data and corpus-based approaches to NLP on Empirical Methods on Natural Language Processing and very large corpora held in conjunction with the Association for Computational Linguistics*, pages 45–52, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[11] Rebecca Hwa. On minimizing training corpus for parser acquisition. In Walter Daelemans and Rémi Zajac, editors, *Proceedings of the 5th Conference on Computational Natural Language Learning (CoNLL-2001)*, pages 84–89. Toulouse, France, 2001.

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

[13] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430, 2003.

[14] Florian Laws and Hinrich Schütze. Stopping criteria for active learning of named entity recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 465–472, Manchester, 2008.

[15] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[16] Teresa Lynn, Jennifer Foster, Mark Dras, and Elaine Uí Dhonnchadha. Active learning and the irish treebank. In *Proceedings of the Australasian Language Technology Association Workshop 2012*, pages 23–32, Dunedin, New Zealand, December 4–6 2012.

[17] Grace Ngai and David Yarowsky. Rule writing or annotation: Cost-efficient resource usage for base noun chunking. In *Proceedings of the Association for Computational Linguistics*, pages 117–125, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[18] Miles Osborne and Jason Baldridge. Ensemble-based active learning for parse selection. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *Proceedings of the Human Language Technology-Conference of the North American Chapter of the Association for Computational Linguistics*, pages 89–96, Boston, Massachusetts, USA, 2004.

[19] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175, 1999.

[20] Burr Settles. *Active Learning: Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers, 2012.

[21] H. Sebastian Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 287–294, New York, NY, USA, 1992. ACM.

[22] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423,623–656, 1948.

[23] Mark Steedman, Rebecca Hwa, Stephen Clark, Miles Osborne, Anoop Sarkar, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. Example selection for bootstrapping statistical parsers. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 157–164, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[24] Andreas Stolcke. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, 2002.

[25] Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. Active learning for natural language parsing and information extraction. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 406–414, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

Auxiliary Fronting in German: A Walk in the Woods

Erhard Hinrichs and Kathrin Beck

Department of General and Computational Linguistics
University of Tübingen
Email: {eh;kbeck}@sfs.uni-tuebingen.de

Abstract

This paper utilizes three linguistically annotated text corpora of contemporary and diachronic German to trace the historical development of the fronting of auxiliaries triggered by so-called substitute infinitive forms. The paper demonstrates the added value that annotated corpora can provide for in-depth studies in historical syntax. At the same time it showcases the added value of interoperable language resources for linguistic investigations that require access to and analysis of multiple linguistic resources.

1 Introduction

The historical development and the linguistic triggering environments for auxiliary fronting in German is a long-standing research question in German linguistics, dating at least as far back as Jacob Grimm's famous *Deutsche Grammatik* [6]. Auxiliary fronting occurs in subordinate clauses as in (1) and involves, inter alia, modal verbs such as *können* and *müssen*.

(1) a. *dass Eike gesungen hat.*
that Eike sung has.
'that Eike has sung.'
b. *dass Eike hat singen {können / müssen}.*
that Eike has sing be able to / have to.
'that Eike was able / had to sing.'
c. * *dass Eike singen {können / müssen} hat.*
that Eike sing be able to / have to has.
d. * *dass Eike kommen {gekonnt / gemusst} hat.*
dass Eike come be able / have to has
e. * *dass Eike hat kommen {gekonnt / gemusst}.*
dass Eike has come be able / have to.

(1a) shows that the finite auxiliary in a German subordinate clause normally appears in clause-final position. However, when the finite auxiliary governs a modal such as *können* or *müssen*, as in (1b), then the finite auxiliary is fronted. The ungrammaticality of (1c) shows that in such cases auxiliary fronting is, in fact, obligatory. Auxiliary fronting appears in conjunction with so-called *Ersatzinfinitiv* (substitute infinitive) forms of past participles for modal verbs such as *können* (instead of the expected past participles *gekonnt*) und *müssen* (instead of the expected *gemusst*) as the ungrammaticality of (1d) and (1e) illustrates. There are at least two considerations that make a diachronic corpus study of the auxiliary fronting construction worthwhile and significant: (i) starting with the work of Jacob Grimm, German linguists have been wondering about the historical development of this construction and the curious interplay between auxiliary fronting and the accompanying substitute infinitive forms, (ii) the empirical generalizations about the range of verbs participating in the fronting construction have been rather unclear and a matter of considerable controversy. In many cases, such empirical generalizations have largely been based on grammaticality judgments of native speakers. Considering corpus data and thereby broadening the empirical evidence appears to be a much-needed extension of previous research. Such corpus evidence has not been available until very recently, due to the unavailability of digital corpora with sufficient amounts of data and linguistic annotations. The availability of such corpora as part of the Common Language Resources and Technology Infrastructure (CLARIN)[1] has made it possible to fill this gap. The present study will make use of three linguistically annotated corpora: the Tübingen treebanks TüBa-D/Z [13] and TüPP-D/Z [10] and the German Text Archive DTA [5] of historical texts hosted at the Berlin-Brandenburg Academy of Sciences (BBAW).

A crucial aspect of the linguistic investigations made possible as part of the CLARIN infrastructure concerns the interoperability of the treebanks and DTA resources mentioned above. Since all resources involved share a common layer of part-of-speech annotation, using the same STTS tag set [11] for German, it becomes possible to search for the same patterns in all resources and thus track linguistic change over more than four centuries.

The remainder of the paper is structured as follows: Section 2 presents a more complete overview of the auxiliary fronting construction. In section 3, the three corpora used in this study will be introduced. Section 4 presents the empirical results of the corpus study and interprets these results in a diachronic perspective. Section 5 summarizes the main results and indicates some directions for future research.

[1] www.clarin.eu

2 Auxiliary Fronting in German

This section gives an overview of the auxiliary fronting construction in German. This overview covers the different tenses in which it occurs as well as the different classes of verbs that can trigger this construction. As mentioned above, the grammaticality judgments about the construction are subject to considerable variation. This variation is mainly attributed to dialectal differences. The grammatical judgments reported in this section follow those reported in [3], which presents the most comprehensive overview of the construction in recent years.

The construction occurs only with coherent infinitive constructions in the sense of Gunnar Bech [1]. Coherent infinitive constructions involve bare infinitives, while incoherent infinitive constructions involve *zu*-infinitives. The examples in (2) illustrate seven different verb classes that enter into the coherent infinitive constructions in the perfect tense in German.[2] The reason for distinguishing these seven classes has to do with the following two factors: (i) whether auxiliary fronting is obligatory or not, and (ii) whether the use of a substitute infinitive is obligatory or whether an 'ordinary' past participle is also licit.

(2) a. *dass sie {* arbeiten gekonnt hat / * arbeiten können hat /*
that she work be able has / work be able to has/
hat arbeiten können}.
has work be able to
'that she was able to work.'

b. *dass sie nicht {? arbeiten gebraucht hat / * arbeiten brauchen hat*
that she not work needed has / work needed has
/ hat arbeiten brauchen}.
/ has work needed
'that she did not need to work.'

c. *dass sie ihn {arbeiten gelassen hat / arbeiten lassen hat /*
that she him work let has / work let has
hat arbeiten lassen}.
has work let
'that she let him work.'

d. *dass sie ihn {arbeiten gesehen hat / arbeiten sehen hat /*
that she him work see has / work see has /
hat arbeiten sehen}.
has work see
'that she saw him work.'

[2] The examples are taken from [3], p. 250

e. *dass sie {arbeiten gelernt hat / * arbeiten lernen hat /*
that she work learnt has / work learnt has /
** hat arbeiten lernen}.*
has work learnt
'that she has learnt how to work.'

f. *dass sie {sitzen geblieben ist / * sitzen bleiben ist /*
that she sit remained is / sit remain is /
** ist sitzen bleiben}.*
is sit remained
'that she remained seated.'

g. *dass sie {arbeiten gegangen ist / * arbeiten gehen ist /*
that she work gone is / work gone is /
** ist arbeiten gehen}.*
is work gone
'that she went to work.'

Apart from modal verbs such as *können*, as in (2a), it concerns the negated form of the auxiliary *brauchen*, as in (2b). In both cases, the fronting of the finite auxiliary is obligatory. For the auxiliary *lassen* and AcI verbs such as *sehen*, as in (2c) and (2d), fronting is optional, and both the substitute infinitive and the past participle forms are admissible. For the remaining cases of coherent infinitive constructions in (2e-2g), auxiliary fronting is ungrammatical, and only the past participle forms are possible.

In addition to the perfect tense, auxiliary fronting also occurs in the pluperfect with forms of *hatten* (instead of *haben* as in (2)) and in the future tense with forms of *werden* as in (3).[3]

(3) a. *dass sie {arbeiten können wird / wird arbeiten können}*
that she work be able to will / will work be able to
'that she will be able to work.'

b. *dass sie nicht {arbeiten brauchen wird / wird arbeiten brauchen}*
that she not work need will / will work need
'that she did not need to work.'

c. *dass sie ihn {arbeiten lassen wird / wird arbeiten lassen}.*
that she him work let will / will work let
'that she will let him work.'

d. *dass sie ihn {arbeiten sehen wird / wird arbeiten sehen}.*
that she him work see will / will work see
'that she will see him work.'

e. *dass sie {arbeiten lernen wird / wird arbeiten lernen}.*
that she work learn will / will work learn
'that she will learn how to work.'

[3] The examples are taken from [3], p. 250

f. *dass sie {sitzen bleiben wird / wird sitzen bleiben}.*
that she sit remain will / will sit remain
'that she will remain seated.'
g. *dass sie {arbeiten gehen wird / wird arbeiten gehen}.*
that she work go will / will work go
'that she will go work.'

The grammaticality judgments presented in (3) differ from those shown in (2) for the perfect tense in two respects: fronting of word forms of *werden* is considered optional in all cases and grammatical for all seven verb classes (see [8] for more discussion).

The examples in (2) and (3) show the data set that is covered by the corpus study presented in section 4 below. There are additional facts about the construction that are exemplified in (4). For reasons of space, they are not within the scope of the present study.

(4) a. *dass er nicht wird länger bleiben können.*
that he not will longer stay be able to
'that he will not be able to stay longer.'
b. *dass er das Examen bestehen wird können.*
that he the exam pass will be able to
'that he will be able to pass the exam.'
c. *dass er den net will komme lasse.*
that he him not wants to come let
'that he does not want to let him come.'

In (4a) the auxiliary *wird* is not the leftmost element in the verbal complex, but is fronted to the left of a preceding adverbial. Example (4b) is an instance of the so-called *Zwischenstellung* (intermediate position), where the finite auxiliary does not occupy the leftmost position in the verbal complex, but rather is placed as the leftmost auxiliary to the right of the main verb. Example (4c) is from [4]; it belongs to the Swabian dialect of German, where the auxiliary fronting construction has been generalized to the verb *wollen*.

Apart from the usage patterns of auxiliary fronting attested for modern German, the origin and the historical development of the construction are a matter of considerable interest and debate. The explanations given in the literature range from morphosyntactic accounts [6] to semantic ones [12]. The hypothesis put forth by Jacob Grimm ([6], p. 168) rests on the assumption of an accidental identity of the infinitival and past participle forms of strong verbs, which had a tendency to drop their prefix *ge-*, thus resulting in past participle forms such as *lassen* (rather than *ge-lassen*) that are identical to the infinitive. Grimm points out that the dropping of the *ge-* prefix is well attested historically for highly frequent strong verbs such as *heißen*, *lassen*, and *sehen*. He considers this finding as supporting evidence for his hypothesis concerning the origin of the construction.

3 The TüBa-D/Z, TüPP-D/Z and DTA corpora

The TüBa-D/Z[4] treebank is an annotated German newspaper corpus based on data taken from the daily issues of 'die tageszeitung' (taz). The TüBa-D/Z currently comprises 75,408 sentences (1,365,642 tokens). Figure 1 illustrates the annotation layers of part-of-speech annotation and syntactic annotation, including the annotation of auxiliary fronting construction.

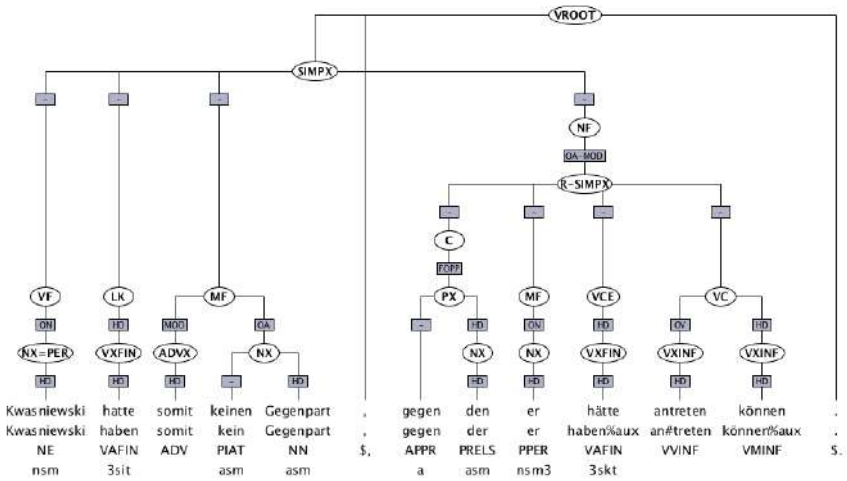


Figure 1: Annotation of auxiliary fronting in the TüBa-D/Z

The terminal nodes in the parse tree are labeled with part of speech tags taken from the STTS tag set. Non-terminal nodes of the tree include maximal projections of lexical categories such as ADVX (adverbial phrase), NX (noun phrase), and PX (prepositional phrase), but also a layer of topological fields such as VF (Vorfeld), LK (Linke Klammer), MF (Mittelfeld), and VC (Verbkomplex). Topological fields in the sense of Höhle [9], Herling [7], and Drach [2] are widely used in descriptive studies of German syntax. Such fields constitute an intermediate layer of analysis above the level of individual phrases and below the clause level. Auxiliary fronting is annotated by a special tag VCE (short for: Verbkomplex mit Ersatzinfinitiv; *verbal complex extension*) that dominates the preterminal label AUX of the fronted auxiliary.

Apart from syntactic phrase labels and topological fields, the syntactic annotation layer of the TüBa-D/Z also includes grammatical function information. Grammatical functions are annotated as edge labels that connect nodes labeled with syntactic categories. Grammatical function labels include HD (short for: head of a phrase) and nominal complement labels ON and OA, OD, and OG (short for: nominative, accusative, dative, and genitive

[4] www.sfs.uni-tuebingen.de/ascl/ressourcen/corpora/tueba-dz.html

complements), MOD (short for modifier) and OA-MOD (short for: modifier of an accusative complement). Table 3.8 on page 18 of the TüBa-D/Z Stylebook [13] presents a complete overview of all grammatical functions that are distinguished in the TüBa-D/Z treebank.

The TüPP-D/Z (Tübingen Partially Parsed Corpus of Written German)[5] corpus uses as its data source the Scientific Edition of the taz German daily newspaper[6], which includes articles from September 2, 1986 up to May 7, 1999. All articles have been automatically annotated with clause structure, topological fields, and chunks, as well as parts of speech and morphological ambiguity classes. All texts are processed automatically, starting from paragraph, sentence, word form, and token segmentation. A more in-depth description of the linguistic annotation can be found in the TüPP-D/Z stylebook [10], and information about the actual XML encoding of linguistic annotation can be found in the TüPP-D/Z markup guide [14]. The TüPP-D/Z corpus does not contain any special markup for the auxiliary fronting construction. On the basis of the topological field annotation in the TüPP-D/Z corpus, instances of the auxiliary fronting construction can, however, be easily found in the corpus with the use of the TIGERsearch query tool.

The German text archive (DTA)[7] contains texts ranging from 1610 to 1900. The texts have been digitized and transliterated, using a high-precision double-keying method. The archive is still under construction. The version used for the present study dates from April 2013 and consists of 65,903,329 word tokens taken from 288,013 digitized pages. The texts represent different genres, including novels and other literary works, scientific and journalistic texts.[8]

The linguistic annotation of the DTA is at the level of individual word forms. It includes lemmatization, part-of-speech information, and spelling normalization. The latter is very important for historical German texts due to the lack of spelling norms in previous centuries. The part-of-speech and lemma information is sufficient to define highly reliable search patterns for the DTA text collection, using the query syntax provided by the online search tool available on the DTA webpage. More information about the query syntax is available online[9].

The interoperability of the linguistic annotation in the three corpora is of utmost importance for the present study. This interoperability is ensured by the use of the same tag set, STTS, in the DTA and in the two treebanks. The syntactic annotation in the treebanks, which is not entirely identical, contains in both cases topological field information, which is crucial for the precision

[5] http://www.sfs.uni-tuebingen.de/ascl/ressourcen/corpora/tuepp-dz.html
[6] www.taz.de
[7] http://www.deutschestextarchiv.de/
[8] More information about the genres represented in the DTA is given at http://www.deutschestextarchiv.de/web/doku/textauswahl.
[9] http://www.deutschestextarchiv.de/doku/DDC-suche_hilfe

and recall of searching for the fronting construction. Precision and recall is, of course, the highest in the case of the TüBa-D/Z treebank since it contains dedicated labels that mark all instances of this construction. The search results are noisier for the other two corpora since the search is heuristic in nature, relying on the word order and morphosyntactic properties of the construction.

4 Corpus Query Results

This section presents the quantitative results of the corpus queries in the three linguistically annotated corpora described in the previous section.

4.1 Auxiliary Fronting for *werden*

Table 1 shows the results for the fronting of the forms of *werden* in the future tense.[10] The presentation of the data follows the order of presentation in example (2) above.

	TüBa-D/Z	TüPP-D/Z	DTA
arbeiten können ***wird***	0	57	3
wird *arbeiten können*	**9**	**1092**	**852**
arbeiten lassen ***wird***	**4**	**476**	187
wird *arbeiten lassen*	2	70	**233**
arbeiten sehen ***wird***	0	**12**	5
wird *arbeiten sehen*	0	0	**16**
arbeiten lernen ***wird***	0	2	**40**
wird *arbeiten lernen*	0	1	**32**
sitzen bleiben ***wird***	0	**43**	**17**
wird *sitzen bleiben*	0	23	12
arbeiten gehen ***wird***	0	9	1
wird *arbeiten gehen*	0	8	2

Table 1: Corpus results for the placement of the auxiliary *werden*

Each row in Table 1 represents the raw counts of the pattern in question in the three corpora. There is a big difference in the number of relevant data points present in the TüBa-D/Z and the other two corpora. The TüBa-D/Z with 1,365,642 tokens only contains examples with modal auxiliaries and with *lassen*. The patterns in question for the remaining five verb classes do not appear at all in the corpus. By contrast, the other two corpora with more than 200 million tokens (TüPP-D/Z) and approx. 66 million tokens (DTA)

[10] In Table 1 and in Table 2 below, the auxiliary in question is rendered in boldface. For each of the seven verb classes, the number of occurrences with the comparatively higher frequency in a particular corpus is also shown in boldface.

are many times larger, leading to a much wider spread of data points across the different verb classes. For both the TüPP-D/Z and the DTA, modal auxiliaries are by far the most frequent class, and the verb *lassen* the second most frequent class.

There is an interesting difference in the relative frequency of fronted auxiliaries and auxiliaries in clause-final position. Fronted *werden* is the preferred pattern for the class of modals. This is true both synchronically (in the taz corpus) and diachronically (in the DTA corpus collection). For *lassen*, verb-final placement is more frequent (by a factor of approx. 7) synchronically (in the taz corpus), while the fronted auxiliary position is slightly more frequent in the diachronic corpus. If one considers the relative frequency of fronted and clause-final *werden* over time, then the following dynamics is evident: historically, fronting occurred with all seven verb classes. The only verb class for which the verb-final position is strongly dispreferred are the modal auxiliaries. Synchronically, there is a clear split between the modals, for which fronted *werden* is much more frequent and all the other verb classes, for which verb-final placement is strongly preferred.

4.2 Auxiliary Fronting for the perfect tense

Table 2 shows the results for the fronting of the forms of *haben* in the perfect tense. The presentation of the data follows the order of presentation in example (3) above.

	TüBa-D/Z	TüPP-D/Z	DTA
hat *arbeiten können*	**62**	**6888**	**5440**
* *arbeiten gekonnt* ***hat***	0	0	0
* *arbeiten können* ***hat***	0	7	0
hat *arbeiten lassen*	14	**1921**	**1412**
arbeiten gelassen ***hat***	0	40	13
arbeiten lassen ***hat***	**38**	17	3
hat *arbeiten sehen*	**62**	**1178**	**233**
arbeiten gesehen ***hat***	0	0	16
arbeiten sehen ***hat***	1	2	0
* ***hat*** *arbeiten lernen*	0	1	114
arbeiten gelernt ***hat***	2	**141**	**540**
* *arbeiten lernen* ***hat***	0	0	0
* ***ist*** *sitzen bleiben*	0	0	0
sitzen geblieben ***ist***	0	**49**	**65**
* *sitzen bleiben* ***ist***	0	0	0
* ***ist*** *arbeiten gehen*	0	0	0
arbeiten gegangen ***ist***	0	**39**	**16**
arbeiten gehen ***ist***	0	1	1

Table 2: Corpus results for the placement of the auxiliary *haben*

As in Table 1, Table 2 shows a striking difference in the number of data points between the TüBa-D/Z and the other two corpora, due to radically different corpus sizes. As in the case of *werden*, modal verbs provide by far the largest number of data points, followed by the verb *lassen*. AcI verbs form the third most frequent class in the TüPP-D/Z, outranking the verb *lernen*. In the DTA, these frequency ranks are reversed. It seems plausible that this difference is due to genre differences between the two corpora. TüPP/D-Z has newspaper articles as its data source, where AcI verbs are expected to occur with higher frequency.

Table 2 confirms that auxiliary fronting is obligatory for forms of *haben* that govern modal verbs, *lassen*, or AcI verbs such as *sehen*, while the clause-final position is used for the remaining verb classes. This is true for both the synchronic data and for the diachronic data of the DTA. The fact that *lassen* and *sehen* are used with high frequency in the DTA and that the ordinary participial form *gelassen* and *gesehen* are hardly attested in the DTA also confirms Grimm's observation that these verbs had a tendency to drop their *ge-* prefixes, thus resulting in an identity of form between the infinitive and the participle. It appears that this usage pattern is firmly established by the 17th century, the earliest period covered by the DTA. In fact, the use of such *Ersatzinfinitive*s has already spread to modal auxiliaries, as the usage patterns for modals in Table 2 shows. This empirical finding is at least consistent with Grimm's view ([6], p. 169) that the construction dates back in time to the 13th/14th century and that it arose as a contact phenomenon with the Dutch language.

It seems appropriate to add some methodological remarks about the nature of the data and about the findings obtained from them. What these corpus data reveal are patterns of usage of the construction in authentic written materials. However, language use and judgments of grammaticality should not be confused or equated with one another. Grammaticality judgments concern both positive data (deemed to be grammatical) and negative data (classified by native speakers to be ungrammatical). Corpus data can only provide tendencies of actual language use. Needless to say, the absence of certain data from large corpora, while not to be equated with data judged ungrammatical by native speakers, is certainly informative and significant for linguistic theorizing.

5 Conclusion

The use of historical and synchronic corpora, which include relevant levels of linguistic annotations, makes it possible to track syntactic constructions across time and to witness syntactic change. The case study of auxiliary fronting in German has shown that massive amounts of data are necessary if the construction is relatively rare. This, in turn, means that data mining of a sufficient amount of instances of such constructions cannot rely exclusively on manually annotated corpora such as the TüBa-D/Z treebank, but will have

to make use of (semi-)automatically annotated corpora such as the TüPP-D/Z corpus. Linguistic annotation of the data sources for data mining is crucial since it is impossible to manually search large data sets. High-quality linguistic annotation makes it possible to define search queries that result in high accuracy of the search results. If several data sources are utilized, as in the case at hand, then interoperability among the annotations present in the individual corpora greatly facilitates a meaningful comparison of the query results obtained from each corpus. In the case at hand, the use of a common tag set (STTS) in all three corpora ensures such interoperability.

The present corpus study, based on the Tübingen treebanks TüBa-D/Z and TüPP-D/Z and on the German Text Archive DTA of historical texts corroborates Grimm's conjecture that the construction originated with past participles of high-frequency strong verbs, which had a tendency to drop their *ge-* prefix. Corpus evidence over more than four centuries also clearly shows that the construction has been generalized over time to the fronting of the auxiliary *werden* for modals. In future work, it would be fascinating and highly worth-while to be able to extend the diachronic investigation of auxiliary fronting further back in time, ideally to the 13th and 14th century, which according to Grimm mark the period when the construction first entered the German language.

References

[1] Bech, Gunnar (1955/57). *Studien über das deutsche verbum infinitum*. (= Det Kongeliege Danske Videnskabers Selskab; Dan. Hist. Filol. Medd. Bind 35, no. 2 (1955) & Bind 36, no. 6 (1957)). Reprint: Niemeyer, Tübingen 1983.

[2] Drach, Erich (1937). *Grundgedanken der Deutschen Satzlehre*. Frankfurt/Main.

[3] Eisenberg, Peter, Smith, George, and Teuber, Oliver (2001). Ersatzinfinitiv und Oberfeld – Ein großes Rätsel der deutschen Syntax. *Deutsche Sprache* 29, pp. 242-260.

[4] Fritz, Gerd (1992). Remarks on the structure of the verbal complex in early 17th century German. In: R. Tracy (Ed.): *Who climbs the grammar-tree*. Tübingen, pp. 53-66.

[5] Geyken, Alexander, Haaf, Susanne, Jurish, Bryan, Schulz, Matthias, Steinmann, Jakob, Thomas, Christian, and Wiegand, Frank (2011). Das Deutsche Textarchiv: Vom historischen Korpus zum aktiven Archiv. In S. Schomburg, C. Leggewie, H. Lobin, and C. Puschmann (Eds.). *Digitale Wissenschaft. Stand und Entwicklung digital vernetzter Forschung in Deutschland, 20./21. September 2010. Beiträge der Tagung*, pp. 157–161.

[6] Grimm, Jacob (1837). *Deutsche Grammatik. Vierter Teil*. Göttingen: Dietterische Buchhandlung.

[7] Herling, Simon Heinrich Adolf (1821). Über die Topik der deutschen Sprache. In *Abhandlungen des frankfurterischen Gelehrtenvereins für deutsche Sprache*, pp. 296–362, 394. Frankfurt/Main. Drittes Stück.

[8] Hinrichs, Erhard W. and Nakazawa, Tsuneko (1994). Linearizing AUXs in German Verbal Complexes. In: J. Nerbonne, K. Netter and C. Pollard (eds.), *German in Head-Driven Phrase Structure Grammar*. Volume 46 of *CSLI Lecture Notes*, pp. 11-37. Stanford, CA.

[9] Höhle, Tilmann N. (1986) Der Begriff 'Mittelfeld'. Anmerkungen über die Theorie der topologischen Felder. In A. Schöne (Ed.) *Kontroversen alte und neue. Akten des 7. Internationalen Germanistenkongresses Göttingen*, pp. 329–340. Tübingen: Niemeyer.

[10] Müller, Frank Henrik (2004). *Stylebook for the Tübingen Partially Parsed Corpus of Written German (TüPP-D/Z)*. Technical Report. Department of General and Computational Linguistics, University of Tübingen, Germany. URL: http://www.sfs.uni-tuebingen.de/tupp/dz/stylebook.pdf

[11] Schiller, Anne, Teufel, Simone, Stöckert, Christine, and Thielen, Christine (1999). *Guidelines für das Tagging deutscher Textcorpora mit STTS*. Technical report, Universities of Stuttgart and Tübingen, Germany. URL: www.sfs.uni-tuebingen.de/fileadmin/static/ascl/resources/stts-1999.pdf

[12] Telljohann, Heike, Hinrichs, Erhard W., and Sandra Kübler (2004). The TüBa-D/Z Treebank: Annotating German with a Context-Free Backbone. *Proceedings of LREC 2004*, pp. 1061-1066.

[13] Telljohann, Heike, Hinrichs, Erhard W., Kübler, Sandra, Zinsmeister, Heike, and Beck, Kathrin (2012). *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*. Department of General and Computational Linguistics, University of Tübingen, Germany. URL: www.sfs.uni-tuebingen.de/fileadmin/static/ascl/resources/tuebadz-stylebook-1201.pdf

[14] Ule, Tylman (2005). *Mark Up Manual for the Tübingen Partially Parsed Corpus of Written German (TüPP-D/Z)*. Technical Report. Department of General and Computational Linguistics, University of Tübingen, Germany. URL: http://www.sfs.uni-tuebingen.de/tupp/dz/markupmanual.pdf

Minimizing Validation Effort for Treebank Expansion

Naman Jain, Sambhav Jain and Dipti Misra Sharma

Language Technologies Research Centre,
International Institute of Information Technology Hyderabad
E-mail: naman.jain@students.iiit.ac.in,
sambhav.jain@research.iiit.ac.in, dipti@iiit.ac.in

Abstract

In this paper, we present our work towards the effective expansion of treebanks by minimizing the human efforts required during annotation. We are combining the benefits of both automatic and human annotation for manual post-processing. Our approach includes identifying probable incorrect edges and then suggesting *k*-best alternates for the same in a typed-dependency framework. Minimizing the human efforts calls for automatic identification of ambiguous cases. We have employed an entropy based confusion measure to capture uncertainty exerted by the parser oracle and later flag the highly uncertain predictions. To further assist human decisions, *k*-best alternatives are supplied in the order of their likelihood. Our experiments, conducted for Hindi, establish the effectiveness of the proposed approach. We exercised label accuracy as a metric to show the effectiveness by increasing it with economically viable manual intervention. This work leads to new directions in the expansion of treebanks by accelerating the annotation process.

1 Introduction

Last decade has witnessed an increasing interest in dependency-based approaches to syntactic parsing of sentences [19]. It has been observed that morphologically rich and free word order languages can be better handled using the dependency based framework than the constituency based one [2].

A major goal of dependency parsing research is to develop quality parsers, that can provide reliable syntactic analysis to various NLP applications such as natural language generation [21], machine translation [22], ontology construction [18], etc. To build such high-quality dependency parsers, there is a certain need of high-quality dependency annotated treebanks for the training and the evaluation phases involved [4]. Since the parsing accuracy largely depends on the amount of labeled training data, annotated data plays a crucial role [7]. The obstacle in building

large sized annotated data is the availability of human experts for the process of annotation.

In past, different approaches had been implemented to address the problem of availability of annotated data. Developing automatically annotated treebanks [20] was one of the proposed solutions. But the disadvantage with this approach is the presence of parsing errors in automatically parsed data. The remedy is to carry out a manual validation step to eradicate the errors in the automatic parsed data. However, validating the whole data is still time consuming and it might be argued that the annotation of whole dataset from scratch, by an expert annotator, would have been a better choice since it will result in much more accurate annotated data with comparable increase in time. Also the increase in time can be justified against the more accurately annotated data which results in better data driven NLP systems. Dickenson et al. in [5, 6] proposed a selective review process where a human validator only reviews the cases highly probable to be erroneous. They identified *ad hoc* rules from the treebank, which are unlikely to be used in general, and later compare these rules on the parsed data. The anomalous part of the parse tree as per the *ad hoc* rules is flagged as potential error which is later subjected to manual validation.

Our approach involves running a data driven dependency parser followed by a selective human validation step. But instead of validating all the edges, validation will be done for the edges, in which the involved parser is highly uncertain during prediction. We have employed an entropy based confusion measure to capture uncertainty exerted by the parser oracle and later flag the highly uncertain predictions. To further assist human decision, we also provide k probable alternatives in the order of their likelihood. In all, the approach comprises of the following two steps:-

- Identification of probable incorrect predictions.
- Selection of k-best alternates.

2 Background

The fundamental notion of dependency is based on the idea that the syntactic structure of a sentence consists of binary asymmetrical relations between the words, termed as dependencies. In a typed dependency framework, the relation between a pair of words, is marked by a *dependency label*, where one of the nodes is *head* and other is *dependent* [10]. Figure 1 shows the syntactic relations[1] between the words of a sentence and the dependency labels marked along the edges.

[1] k1: Doer, k1s: Noun Complement, k3: Instrument, k5: Source, k7p: Place, k7t: Time, pof: Part-of (complex predicate), nmod: Noun Modifier, lwg__psp: Local-word-group postposition

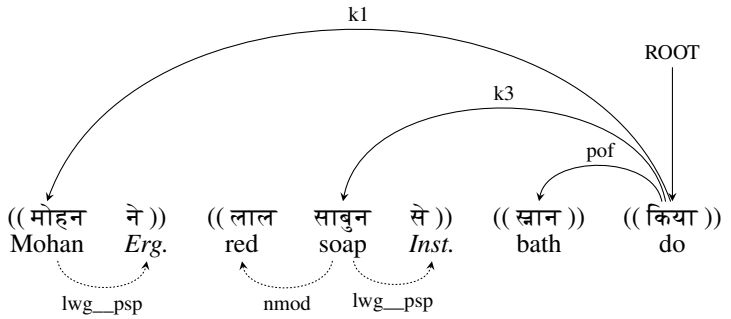


Figure 1: Example of Dependency Tree with syntactic relations.

We have worked with Hindi, a relatively free-word-order and morphologically rich, Indo-Aryan language. In previous attempts to parse Hindi[1], it has been observed that UAS[2] is greater than LS[3] by $\sim$ 6 percentage points, which is reconfirmed by our baseline parser (later described in Section 4) where UAS is 6.22% (UAS - LS) more than LS. The UAS in our baseline is well above 90% (92.44%) while the LS is still 86.21%. This drives us to focus on improving LS, to boost the overall accuracy(LA[4]) of the parser.

Dependency annotation scheme followed in Hindi Dependency Treebank [3] consists tag-set of $\sim$ 95 dependency labels which is comparatively larger than the tag-set for other languages[5], like Arabic($\sim$ 10), English($\sim$ 55), German($\sim$ 45) etc. This apparently is a major reason behind the observed gap between LS and UAS for Hindi parsing. One of the frequent labeling errors that the parser makes is observed to be between closely related dependency tags, for eg. $k7$ (abstract location) and $k7p$ (physical location) are often interchangeably marked [17]. We have reasons to believe that such a decision is comparatively tougher for an automatic parser to disambiguate than a human validator.

In the past, annotation process has benefited from techniques like Active Learning [14] where unannotated instances exhibiting high confusions can be prioritized for manual annotation. However, in Active Learning, the annotators or validators generally have no information about the potentially wrong sub-parts of a parse and thus full parse needs to be validated. Even if the the annotators are guided to smaller components (as in [15]), the potentially correct alternates are not endowed. In our approach the validator is informed about the edges which are likely to be incorrect and to further assist the correction k best potential label-replacements are also furnished. So, effectively just partial corrections are required and only in worst case (when a correction triggers correction for other nodes also) a full sentence needs to be analyzed.

[2]UAS = Unlabeled Attachment Score

[3]LS = Label Accuracy Score

[4]LA = Labeled Attachment Score

[5]As observed on the CoNLL-X and CoNLL2007 data for the shared tasks on dependency parsing.

The efforts saved in our process are tough to be quantified, but the following example provides a fair idea of efficacy of our proposition. In figure 2, second parse has information of the probable incorrect label and also has 2 options to correct the incorrect label to guide a human validator.

(1) ... जेलों में समस्या हल करने ...
... prisons in problems solve do ...
... solve problems in prisons ...

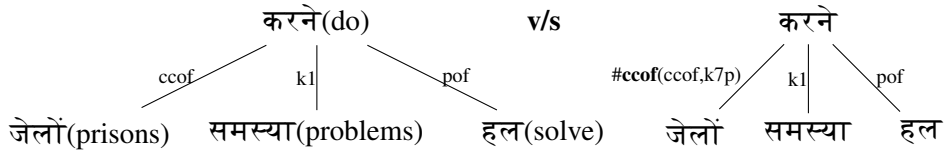


Figure 2: Example showing output from conventional parser v/s output from our approach. Arc-label with '#' represents incorrect arc label (confusion score $> \theta$) along with 2-best probable arc labels.

3 Methodology

The second sub-figure in figure 2 shows a typical output from our approach. The overall methodology, to obtain such an output, is divided into two sequential steps:-

1. Predicting Incorrect Labels: First we intend to predict and flag potentially incorrect labels. Some recent efforts have tried to extend the functionality of the parser with additional information on the quality of the output parse. Mejer and Crammer [12] have worked with MSTParser [11] to give confidence scores for attachments while Jain and Agrawal [8] have worked with MaltParser [13] to render the confusion scores for arc-labels. Since our focus is on arc-labels we follow the approach proposed in [8]. They captured the confusion exerted on the parser's oracle while predicting a parser action and propagated it to the arc-label of the dependency tree. The quantification of confusion is done by calculating entropy with the class membership probabilities of the parser actions.

We obtained the confusion score for each arc-label in our data. Next, we obtained a threshold ($\theta = 0.137$) for which the maximum F_1-*score* is observed for incorrect label identification on the development set(Figure 3). In figure 2, the edge with the label 'ccof' has been flagged (#) because the confusion score is greater than θ, which signifies that it is probably incorrect. The proposition is indeed correct as the correct label is 'k7p' instead of 'ccof'.

The additional details about the correctness of an arc-label, can duly indicate the cases where the probability of the arc-label to be incorrect is high. In our efforts

to minimize the human intervention, we propose to subject the reviewer only to those cases where the confusion score is above θ. At this stage (i.e. without step 2) the reviewer will be required to judge *'if the flagged label is indeed incorrect'* and if it is, then choose the corresponding correct label among all the remaining labels.

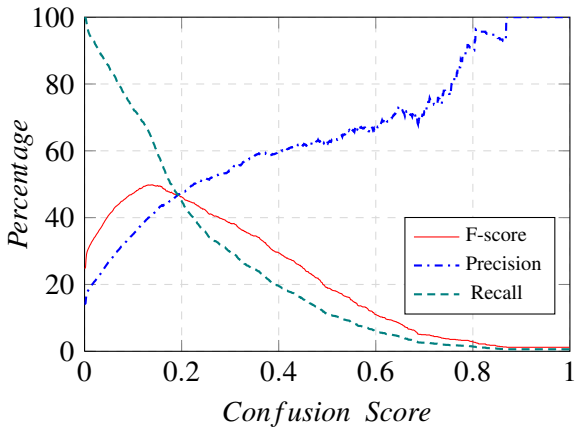


Figure 3: *Precision*, *Recall* and F_1-*score* for various values of confusion score on 'Hindi' development set.

2. *k*-Best Dependency Labels for the Flagged Arc-Labels: To further assist human decision, we also provide k probable alternatives in the order of their likelihood as proposed by the oracle. The reason behind this hypothesis is that it is likely that the correct label exists among the top label candidates. This, potentially, can give quick alternates to the reviewer for choosing the correct label and thereby speedup the review process.

The likelihood of the arc-labels is obtained and ranked using the following three strategies:-

- *Voting*: The list of predicted labels, using voting mechanism, is sorted in decreasing order of number of votes from ${}^nC_2$ binary classifiers, obtained during classification. We choose the top k labels from this sorted list as the k-best alternate labels.

- *Probability*: The calculation of confusion scores demand for class membership probabilities for arc-label (refer step 1). The posterior probabilities for the candidate labels can also be alternatively used to emit out the resultant dependency label. Similar to voting scheme, the labels are sorted in decreasing order of their probabilities. The sorted list of predicated labels may differ from that of *voting* mechanism, which motivate us to also consider *probability* for choosing the k-best dependency labels.

- *Voting* + *Probability*: A tie can occur between two or more labels in the list of k-best candidate labels if their votes/posterior probabilities are same.

However, the phenomenon is unlikely in case of probabilities due to the real valued nature calculated up-to 10 decimal places. On the other hand *votes* are integer-values ($\{0, ..., {}^{n}C_2\}$, where n is number of labels) and are much more susceptible to ties. The tie in voting can be resolved using complement information from probabilities (and vice-versa).

4 Experiments

In our experimental setup, we assume the availability of a human expert for validation of the machine parsed data, who, when queried for a potential incorrect edge label, responds with the correct edge label. The experiments aim to measure the assistance provided to human expert by our approach. We varied the list of k-best labels from k=1 to k=5.

We focus on correctly establishing dependency relations between the chunk[6] heads which we henceforth refer as *inter-chunk* parsing. The relations between the tokens of a chunk (*intra-chunk* dependencies) are not considered for experimentation. The decision is driven by the fact that the *intra-chunk* dependencies can easily be predicated automatically using a finite set of rules [9]. Moreover we also observed the high learnability of *intra-chunk* relations from a pilot experiment. We found the accuracies of *intra-chunk* dependencies to be more than 99.00% for both LA and UAS.

We trained a parser model on the lines of [17] with minor modifications in the parser *feature model*. We employ MaltParser version-1.7 and Nivre's Arc Eager algorithm for all our experiments reported in this work. All the results reported for overall parsing accuracy are evaluated using *eval07.pl*[7]. We use MTPIL [16] dependency parsing shared task data. Among the features available in the FEATS column of the CoNLL format data, we only consider *Tense, Aspect, Modality (tam)*, *postpositions(vib)* and *chunkId* while training the parser. Other columns like POS, LEMMA, etc. are used as such.

In case of typed-dependency parsing, the accuracy can be LA, UAS or LS. However, in our case, since we are focusing on the correct prediction of arc-labels, the results are on LS. In terms of strategies mentioned in Section 3, a baseline system is generated using *Voting* strategy with $k = 1$. The LS is 86.21% as shown in Table 1.

5 Evaluation and Analysis

We wish to evaluate the assistance provided to a human validator by our approach. However, evaluation of interactive parse correction is a complicated task due to

[6] A chunk is a set of adjacent words which are in dependency relation with each other, and are connected to the rest of the words by a single incoming arc.

[7] http://nextens.uvt.nl/depparse-wiki/SoftwarePage/#eval07.pl

intricate cognitive, physical and conditional factors associated with a human annotator. The extent of benefit will differ from annotator to annotator and thus quantifying the gain is a challenging task. We perform two kinds of evaluation to illustrate the effectiveness of our approach.

The first is an automatic evaluation which assumes a perfect reviewer who always identifies incorrect label and picks the correct label from the available k-best list, if correct label is present in the list. Though this would be an ideal scenario but gives the upper bound of the accuracies that can be reached with our approach. The simulation of the perfect reviewer is done using the gold annotation. It is also assumed that the decision of the correct label can be taken with the information of local context and the whole sentence is not reviewed(which is not always true in case of a human annotator). The second is a human evaluation where two annotators are asked to review the arc labels, *with* and *without* the additional information from our system. The gain is measured in form of time saved with the additional information.

5.1 Automatic Evaluation

In our test set, we found $\sim$23% (4,902 edges) of total (21,165) edges having confusion score above θ and thus marked as potentially incorrect arc-labels. Table 1 exhibits LS improved by perfect human reviewer(simulated via gold data), for k-best experiments where k=1 to 5 on $\sim$23% potentially incorrect identified arc-labels.

k	Voting(%)	Probability(%)	Voting+Probability(%)
1	86.21	**86.35**	86.28
2	90.86	**90.96**	90.91
3	92.13	**92.24**	92.18
4	92.72	**92.86**	92.74
5	92.97	**93.16**	93.04

Table 1: k-Best improved LS on inspecting $\sim$23% ($> \theta$) edges.

Table 1 also depicts that as the value of k increases, the label accuracy also increases. The best results are obtained for *Probability* scheme. There is a substantial increment in LS moving from 1-best to 2-best in all the schemes. The amount of gain, however, decreases with increase in k.

Ideally to achieve maximum possible LS, all the edges should be reviewed. Table 2 confirms that if all the edges are reviewed, an LS of 93.18% to 96.57% is achievable for k, ranging over 2 to 5. But practically this would be too costly in terms of time and effort. In order to economize, we wish to only review the cases which are probable enough to be incorrect. Confusion scores give a prioritized list of edges, which dictates the cases that should be dispatched first for review. To relate the review cost against LS gain we present a metric AGI_x defined as:-

AGI_x : *"Accuracy Gain on Inspecting top x% edges"* corresponds to the ratio of accuracy gain from baseline by inspecting top x% of total edges, when sorted

in decreasing order of their *confusion score*. The metric takes into account the human effort that goes into validation or revision, and thus gives a better overview of ROI(Return on Investment).

$$AGI_x = \frac{\text{Accuracy after validating top } x\% \text{ edges} - \text{Baseline accuracy}}{x}$$

From Table 1 and Table 2 we observe for $k = 2$ and *probability* scheme that the improved LSs are 90.96% and 93.18% on inspecting 23% and 100% edges respectively. Although the latter is greater than former by $\sim 2\%$ but this additional increment requires an extra inspection of additional $\sim$77% edges, which is economically inviable. The fact is better captured in Table 3, where AGI_{23} subdues AGI_{100} for different values of k using different 'schemes'.

Further to incorporate the fact that *'larger the candidate list more will be the human efforts required to pick the correct label'*, we also present the results of $\mathbf{AGI_x/k}$, which can govern the choice of k, best suited in practice. While taking this into account, we assume that the human efforts are inversely proportional to k. Results for $\mathbf{AGI_{23}/k}$ on improved LS, over all the experiments are reported in Table 4.

k	Voting(%)	Probability(%)	Voting+Probability(%)
1	86.21	**86.35**	86.28
2	93.19	93.18	**93.26**
3	95.10	95.11	**95.14**
4	95.94	**96.06**	95.97
5	96.41	**96.57**	96.49

Table 2: k-Best improved LS on inspecting 100% edges.

As shown in Table 3, AGI_{23} increases with increase in the value of k, but it is practically inefficient to keep large value of k. Optimum choice of k is observed to be 2 from the metric AGI_x/k, as shown in Table 4, where the maximum value for $\mathbf{AGI_{23}/k}$ is ~ 0.10 for all the 'schemes', which corresponds $k = 2$.

	Voting		Probability		Voting+Probability	
k	$\mathbf{AGI_{23}}$	$\mathbf{AGI_{100}}$	$\mathbf{AGI_{23}}$	$\mathbf{AGI_{100}}$	$\mathbf{AGI_{23}}$	$\mathbf{AGI_{100}}$
1	0.0000	0.0000	**0.0060**	0.0014	**0.0030**	0.0007
2	**0.2008**	0.0698	**0.2051**	0.0697	**0.2029**	0.0705
3	**0.2556**	0.0889	**0.2604**	0.0890	**0.2578**	0.0893
4	**0.2811**	0.0973	**0.2871**	0.0985	**0.2820**	0.0976
5	**0.2919**	0.1020	**0.3001**	0.1036	**0.2949**	0.1028

Table 3: $\boldsymbol{AGI_{23}}$ and $\boldsymbol{AGI_{100}}$ for k=1 to 5

k	$\mathbf{AGI_{23}}$/k (Voting)	$\mathbf{AGI_{23}}$/k (Probability)	$\mathbf{AGI_{23}}$/k (Voting+Probability)
1	0.0000	0.0060	0.0030
2	**0.1004**	**0.1025**	**0.1015**
3	0.0852	0.0868	0.0859
4	0.0703	0.0718	0.0705
5	0.0584	0.0600	0.0590

Table 4: $\boldsymbol{AGI_{23}/k}$ for k=1 to 5

From the above analysis, we can establish that with 2 probable alternatives, a perfect human oracle can increase the LS by 4.61%, inspecting top $\sim$ 23% of total edges. The corresponding LA increase is 4.14%(earlier 83.39% to now 87.53%).

5.2 Human Evaluation

The human evaluation is performed with the assistance of two human experts who have prior experience in the dependency annotation for Hindi. First, the relative proficiency of the annotators is estimated by taking the ratio of the time taken by each in annotating the labels for a small set of 20 sentences. The time taken by annotator A and annotator B is 105 and 80 minutes respectively and thus the relative annotation proficiency of annotator A to B is $105/80$. The disagreement between the two annotators is observed to be less than 3% and hence neglected for the relative proficiency calculation.

Next, both the annotators are provided with six set of 10 parse trees each and asked to review a specific label in each parse. The six sets are formed by picking parse tree corresponding to the nodes having maximum confusion score. Evey parse tree, provided for review, already has the attachments and the labels annotated except for the label under review. Each set has same sentence but one version of the set also has the k-best labels corresponding to the label under review which is alternatively provided to each reviewer. During the review, time is noted for each annotator for each set. The first six rows of the table 5 illustrate the time taken by each reviewer for these six sets. The value of k is taken to be 2,3 and 5 with two sets corresponding to each k. Among these two sets once the k-best labels are provided to Annotator-A and once to Annotator-B, as shown in the second column of table 5.

The impact of the additional information provided in form of k-best labels is quantified in terms of the time gain. Time gain is the difference between the expected time an annotator is supposed to take and the time actually taken with k-best labels information. The expected time is estimated using the relative proficiency calculated earlier. For example, in the first set (first row of table 5) annotator A is provided with the k-best label version of the set while annotator B has no such information in her version of the set. Time taken by annotator B in the review process is recorded to be 15 minutes. The expected time for annotator A can be estimated with the relative proficiency which amounts to 19.69 minutes ($\frac{15\times105}{80}$).

The gain in the time is 9.69 minutes, which is the difference between the expected time (19.69 minutes) and the actual time taken (10 minutes) by annotator A.

Position	Annotator with info. of the k-best labels	k	Time Taken		Expected Time		Gain
			A	B	A	B	
Top	A	2	10	15	19.69	-	**9.69**
Top	B	2	15	11	-	11.43	**0.43**
Top	A	3	15	13	17.06	-	**2.06**
Top	B	3	10	10	-	7.62	-2.38
Top	A	5	10	9	11.81	-	**1.81**
Top	B	5	10	8	-	7.62	-0.38
Middle	A	2	7	7	9.19	-	**2.19**
Middle	B	2	6	9	-	4.57	-4.43
Bottom	A	2	9	11	14.44	-	**5.44**
Bottom	B	2	7	7	-	5.33	-1.67

Table 5: Human evaluation with time gain due to the k-best labels. All time values are in minutes.

The results from first six sets illustrate maximum gain for $k = 2$. So, for $k = 2$ we further chose four more sets taken respectively from the middle and bottom of the top 23% nodes having confusion score above θ. All the results are shown in table 5.

In human evaluation we observed cases where the gain is negative and it typically happens for annotator B. The discussion with the annotators concluded that if the annotator is highly skilled, the extent of aid from our approach may be less as they can directly recall the label by observing the parent and child nodes. However, they further added that the approach would certainly assist semi-proficient, less skilled or relatively novice annotators. The argument can be supported with the fact that annotator B is highly proficient as she took only 80 minutes in comparison to annotator A who took 105 minutes to annotate the same set of parse trees with labels.

6 Conclusion

In this paper we explored the possibility of minimizing human efforts in validation which results in speeding up the annotation process as validation process. A major hurdle in the process is to effectively utilize the costly human resources. We employed an entropy based confusion measure to capture uncertainty exerted by the parser oracle and later flag the highly uncertain labels. We established that with 2 probable alternatives, a human expert can increase the LS by $\sim 6\%$ (approximately 50% error reduction), by inspecting $\sim 23\%$ of total edges. We further asserted that our approach can be very useful in quick treebank expansion and also minimizing the validation effort for treebank expansion. In future we would also like to study the effectiveness of our approach on attachment correction.

Acknowledgement

We would like to thank Preeti Pradhan and Nandini Upasani for helping us in the human evaluation process.

References

[1] Bharat Ram Ambati, Samar Husain, Sambhav Jain, Dipti Misra Sharma, and Rajeev Sangal. Two methods to incorporate local morphosyntactic features in hindi de-pendency parsing. In *The First Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010)*, page 22, 2010.

[2] A. Bharati, V. Chaitanya, R. Sangal, and KV Ramakrishnamacharyulu. *Natural Language Processing: A Paninian Perspective*. Prentice-Hall of India, 1995.

[3] R. Bhatt, B. Narasimhan, M. Palmer, O. Rambow, D.M. Sharma, and F. Xia. A multi-representational and multi-layered treebank for Hindi/Urdu. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 186–189. Association for Computational Linguistics, 2009.

[4] Sabine Buchholz and Erwin Marsi. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164. Association for Computational Linguistics, 2006.

[5] Markus Dickinson. Detecting errors in automatically-parsed dependency relations. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 729–738. Association for Computational Linguistics, 2010.

[6] Markus Dickinson and Amber Smith. Detecting dependency parse errors with minimal resources. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 241–252. Association for Computational Linguistics, 2011.

[7] Daniel Gildea. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202, 2001.

[8] Sambhav Jain and Bhasha Agrawal. A dynamic confusion score for dependency arc labels. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1237–1242, Nagoya, Japan, October 2013. Asian Federation of Natural Language Processing.

[9] Prudhvi Kosaraju, Samar Husain, Bharat Ram Ambati, Dipti Misra Sharma, and Rajeev Sangal. Intra-chunk dependency annotation: expanding Hindi inter-chunk annotated treebank. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 49–56. Association for Computational Linguistics, 2012.

[10] Tesnière Lucien. Eléments de syntaxe structurale. *Kliencksieck, Paris*, 1959.

[11] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics, 2005.

[12] Avihai Mejer and Koby Crammer. Are you sure?: confidence in prediction of dependency tree edges. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 573–576. Association for Computational Linguistics, 2012.

[13] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kubler, Svetoslav Marinov, and Erwin Marsi. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95, 2007.

[14] Miles Osborne and Jason Baldridge. Ensemble-based active learning for parse selection. In *HLT-NAACL*, pages 89–96, 2004.

[15] Manabu Sassano and Sadao Kurohashi. Using smaller constituents rather than sentences in active learning for japanese dependency parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 356–365, 2010.

[16] Dipti Misra Sharma, Prashanth Mannem, Joseph vanGenabith, Sobha Lalitha Devi, Radhika Mamidi, and Ranjani Parthasarathi, editors. *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages*. The COLING 2012 Organizing Committee, Mumbai, India, December 2012.

[17] Karan Singla, Aniruddha Tammewar, Naman Jain, and Sambhav Jain. Two-stage approach for hindi dependency parsing using maltparser. *Training*, 12041(268,093):22–27, 2012.

[18] Rion Snow, Daniel Jurafsky, and Andrew Y Ng. Learning syntactic patterns for automatic hypernym discovery. *Advances in Neural Information Processing Systems 17*, 2004.

[19] Reut Tsarfaty, Djamé Seddah, Sandra Kübler, and Joakim Nivre. Parsing Morphologically Rich Languages: Introduction to the Special Issue. *Computational Linguistics*, 39(1):15–22, 2013.

[20] Gertjan van Noord and Gosse Bouma. Parsed corpora for linguistics. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 33–39. Association for Computational Linguistics, 2009.

[21] Stephen Wann, Mark Dras, Robert Dale, and Cécile Paris. Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 852–860. Association for Computational Linguistics, 2009.

[22] Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. Using a dependency parser to improve smt for subject-object-verb languages. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 245–253. Association for Computational Linguistics, 2009.

Corpus databases with feature pre-calculation

Erwin R. Komen

Radboud University Nijmegen // SIL-International
E-mail: E.Komen@Let.ru.nl

Abstract

Reliably coded treebanks are a goldmine for linguistics research. Answering a typical research question involves: (a) querying a treebank to extract sentences containing the feature to be investigated, (b) recognizing and keeping track of characteristics that determine the way in which the linguistic feature is encoded, and (c) using statistics to find out which (combination) of these characteristics determines the outcome of the linguistic feature. While sufficient tools are available for steps (a) and (c) in this process, step (b) has not received much attention yet. This paper describes how the programs "Cesax" and "CorpusStudio" can be used jointly to construct a "corpus research database", a database that contains the sentences of interest selected in step (a), as well as user-definable pre-calculated characteristics for step (b).

1 Introduction

Research into variation and change of syntactic constructions often consists of (1) automatically finding examples of the construction in a reliably coded treebank, (2) adding characteristics (features) to each of the examples, (3) gathering the results into a database, (4) manually editing the examples in the database, and (5) preparing the list of examples and their features for further statistical work with programs like "R" or "SPSS". The programs "Cesax" and "CorpusStudio" provide a windows-oriented relatively user-friendly way of achieving these goals [7].[1]

CorpusStudio facilitates queries written in the Xquery language [1], taking *xml* encoded treebank texts as input.[2] The program allows each "hit" to be accompanied by a user-definable number of features, and these features can be programmatically calculated, predicted, or given a default value. The results of a query project (which may involve multiple cascaded queries), together with the calculated features, can be saved as an *xml* database. The Cesax program is equipped with a feature to load such databases and contains an editor to work with the examples and their features. Cesax automatically adds a "Notes" field and a "Status" field to each database entry, allowing the user to annotate the database and to keep track of progress made. The database entries come with a predefined preceding and following context, as well as with the treebank syntax. Double-clicking an entry results in jumping

[1] CorpusStudio and Cesax are freely available from http://erwinkomen.ruhosting.nl.

[2] The *xml* format CorpusStudio deals with best is a TEI-P5 derivative using embedded hierarchy [11]. Labelled bracketing treebank files can be imported and transformed into this format using Cesax. CorpusStudio also allows working directly with the Negra and the Alpino formats, but the database features are not (yet) available for them. Future plans include conversion options for these formats.

to the actual location in the corpus file, which helps quickly looking for the larger context when this is needed (it is this simple feature that is perhaps most valued by the users). Cesax also allows exporting the database for use in statistics.

This paper provides a walk through the process described above, and it does so by taking the "progressive inversion" as an example.

2 The progressive inversion

The progressive inversion construction is a subtype of VP inversion [12]. It is similar to the locative inversion, except that the first constituent is a participle clause instead of a prepositional phrase, as for example (1a):

(1) a. [$_{IP\text{-}PPL}$ Trending away on either side of the port] was [$_{NP\text{-}SBJ}$ a bold rocky coast, varied here and there with shingly and sandy beaches].
[fayrer-1900:54]

b. ?[$_{Sbj}$ A bold rocky coast] was trending away on either side of the port.

The uninverted variant of (1a) would be (1b), but the question mark indicates that this is not quite okay for native speakers. The linguistic question I would like to posit for the sake of this walk-through is: "Which features could determine the appearance of a progressive inversion?"

3 Automatically finding examples

Having defined the research question, step (1) in the process of answering it (see Introduction) is to define a query that automatically locates the necessary examples of the linguistic feature that is being targeted. Sentences that contain a progressive inversion need to have the following three elements:

1) Subject
2) Finite verb
3) Participle

Once sentences containing these three elements are located, the order of these elements will show whether an inversion construction is being used (participle-finite verb-subject) or some other construction (such as: subject-finite verb-participle). The task of locating sentences and determining whether they contain a progressive inversion or not can be accomplished in CorpusStudio by using the Xquery code in (2).[3]

What the code does is: select main clauses into variable `$search` (line 2), put the subject of the main clause into `$sbj` (line 5), put the finite verb of the main clause into `$vfin` (line 8), any participle of the main clause is put into `$ptcp` (line 11), determine the word order (line 14,15), return this clause if all the elements are there (line 18-23). The result of running the Xquery code (2) consists of all the sentences containing the required elements for the

[3] The code makes use of standard Xquery functionality (`for-let-where-return`, `if-then-else`, the function "`exists()`"), some built-in Xquery functions ("`ru:matches`", "`ru:relates`", "`ru:back`"), user-defined functions that are elsewhere in the code ("`tb:SomeChildNo`", "`tb:SomeChild`"), and user-defined global variables ("`$_matrixIP`", "`$_subject`").

progressive inversion (subject, finite verb, participle), and these sentences are divided over the word orders 'Ptcp-Vfin-S' and 'Other'.

(2) *Xquery code to find the inversion examples*

```
1.    (: Look in all main clauses :)
2.    for $search in //eTree[ru:matches(@Label, $_matrixIP)]
3.
4.    (: There must be a subject and a finite verb :)
5.    let $sbj := tb:SomeChildNo($search, $_subject, $_nosubject)
6.    let $vfin := tb:SomeChild($search, $_finiteverb)
7.
8.    (: There must be a progressive or ptcp, but not an absolute :)
9.    let $ptcp := tb:SomeChildNo($search,
                                    'IP-PPL*|[VB]AG*|PTP*', '*ABS*')
10.
11.   (: Find out word order :)
12.   let $order := if ( ($vfin << $sbj) and ($ptcp << $vfin))
13.                 then 'Ptcp-Vfin-S' else 'Other'
14.
15.   (: Check conditions: subject, V-fin, progressive, word order :)
16.   where (  exists($sbj)
17.            and exists($vfin)
18.            and exists($ptcp)  )
19.
20.  (: Return the main clause, subcategorize on word order :)
21.  return ru:back($search, '', $order)
```

While the Xquery code in (2) serves its purpose well, a few extensions are required that will show up later in the code. Two particular main clause types need to be excluded, since they skew the data: the quotations (`QTP` clauses) and main clauses with left dislocations (those with an `LFD` element); the algorithm should only look for non-empty subjects.

4 Adding features to the results

Step (2) in the process of addressing the linguistic question at hand (see Introduction) is to add characteristics, or 'features', to each of the examples we find. One way to do this in Xquery is to make a user-defined function. This function, which will receive the name `tb:ProgrInv()`, is called in line 23 of the extended version of the main query (3). The main query is also extended with a test for the exclusion of left-dislocated and quotative main clause type in lines 5-6 and 26, while lines 10 and 27 make sure that empty subjects (such as traces and dislocation markers) are excluded from consideration.

(3) *Add features to the progressive inversion*

```
1.    (: Look in all main clauses :)
2.    for $search in //eTree[ru:matches(@Label, $_matrixIP)]
3.
4.    (: Some clauses need to be excluded :)
5.    let $clsOk := not(exists($search/child::eTree
6.                             [ru:matches(@Label, 'QTP*|*LFD*')]))
7.
8.    (: There must be a subject :)
9.    let $sbj := tb:SomeChildNo($search, $_subject, $_nosubject)
10.   let $sbjOk := not(exists($sbj[child::eLeaf/@Type = 'Star']))
11.
12.   (: There must be a finite verb :)
13.   let $vfin := tb:SomeChild($search, $_finiteverb)
14.
15.   (: There must be a progressive or ptcp, but not an absolute :)
16.   let $ptcp := tb:SomeChildNo($search,
17.                                 'IP-PPL*|VAG*|BAG*|PTP*', '*ABS*')
18.
19.   (: Prepare subcategorization: ptcp type  :)
20.   let $cat := ru:cat($ptcp, 'phrase')
21.
22.   (: Combine features into a CSV for database creation  :)
23.   let $db := tb:ProgrInv($sbj, $vfin, $ptcp)
24.
25.   (: Check conditions: sbj, Vfin, progressive and word order :)
26.   where (  $clsOk
27.            and exists($sbj) and $sbjOk
28.            and exists($vfin)
29.            and exists($ptcp)
30.         )
31.
32.  (: Return clauses found, subcategorize on the word order :)
33.  return ru:back($search, $db, $cat)
```

The function `tb:ProgrInv()` is defined in such a way, that it returns a string array of the features. These features are subsequently passed on to the CorpusStudio engine through the `$db` variable as an argument of the built-in `ru:back()` function, where they will be available for the next step in the process.

Turning now to the feature calculation, there are two kinds of features the database should be equipped with: those that are going to be used for statistics (such as the kind of verb used, the size of the subject), and those that are important for visual inspection by the database user (such as the text of the subject, finite verb and participle). The code for the `tb:ProgrInv()` function where the features are calculated is provided in (4).

(4) *Xquery code that calculates the feature values for one example*

```
1.  (: ------------------------------------------------------------
2.     Name : tb:ProgrInv
3.     Goal : Provide features for the progressive inversion database
4.     History:
5.     13-06-2013     ERK    Created
6.     ------------------------------------------------------------ :)
7.  declare function tb:ProgrInv(
8.     $sbj as node()?, $vfin as node()?, $ptcp as node()?)as xs:string
9.  {
10.  (: ============================================================
11.       Feature calculation starts here
12.       ============================================================ :)
13.   (: Feature #1-3: the text of the ptcp, V-finite and subject :)
14.   let $ptcpText := replace(tb:Sentence($ptcp), ';', ' ')
15.   let $vfinText := replace(tb:Sentence($vfin), ';', ' ')
16.   let $sbjText := replace(tb:Sentence($sbj), ';', ' ')
17.
18.   (: Feature #4: word order -- Ptcp-Vfin-S, or other? :)
19.   let $order := if ( ($vfin << $sbj) and ($ptcp << $vfin))
20.                 then 'Ptcp-Vfin-S' else 'Other'
21.
22.   (: Feature #6: the type of participle :)
23.   let $ptcpType := if (ru:matches($ptcp/@Label,
24.              'IP-PPL*|VAG*|BAG*|PTP*')) then 'Present' else 'Past'
25.
26.   (: Feature #7: the number of constituents after V-finite :)
27.   let $postVf := count($vfin/following-sibling::eTree[
28.                     not(ru:matches(@Label, $_ignore_nodes_conj))])
29.
30.   (: Feature #8: the number of words in the subject :)
31.   let $sbjSize := count($sbj/descendant::eLeaf[@Type = 'Vern'])
32.
33.   (: Feature #9: NPtype of the subject :)
34.   let $sbjType := ru:feature($sbj, 'NPtype')
35.
36.   (: Feature #10: estimate of referentiality of the subject :)
37.   let $sbjRef := ru:RefState($sbj)
38.
39.   (: ============================================================
40.      Combine features into a CSV for database creation
41.      ============================================================ :)
42.   return concat($ptcpText, ';',
43.      $vfinText, ';',      $sbjText, ';',
44.      $order, ';',         $ptcp/@Label, ';',
45.      $ptcpType, ';',      $postVf, ';',
46.      $sbjSize, ';',       $sbjType, ';',
47.      $sbjRef )
48. } ;
```

As far as the features necessary for visual inspection, the function `tb:ProgrInv()` calculates the text of the participle (line 15), the text of the finite verb (line 18) and the text of the subject (line 21).

Statistically important is the dependent variable `$order` as calculated in lines 24-25: this feature either has the value “Ptcp-Vfin-S”, in which case the example is a progressive inversion, or it has the value “Other”, in which case the example is *not* an inversion. The features numbered 5-10 in the Xquery code (4) are independent variables that could all possibly influence the word order, and they are summarized in Table 1.

#	Feature	Explanation
5	PtcpLabel	syntactic label of the participle (VAG, IP-PPL etc)
6	PtcpTense	progressive is 'Present' or 'Past' tense[4]
7	PostVfNum	number of sibling-constituents following V_{finite}
8	SbjSize	number of words in the subject
9	SbjType	NPtype of the subject
10	SbjRef	Estimate for subject's referential status

Table 1 Features that represent independent variables in a statistic analysis

The features numbered 5-8 are 'fixed' in the sense that they are calculated automatically and do not need manual correction. This is not the case for features 9 (SbjType) and 10 (SbjRef). These features are estimated automatically, but they may need manual correction.

The "SbjType" feature, for instance, makes use of the "NPtype" feature that has been added to the original Treebank texts. But this feature has not been determined for some of the Noun Phrases, which are distinguishable by having the feature value "unknown".

The "SbjRef" feature makes use of the built-in CorpusStudio function "`ru:RefState`", which has a success rate of approximately 85% in determining the referentiality of an NP. The values of this feature all need to be checked manually![5]

5 Making a database

Next in the process of a full-fledged linguistic analysis as mentioned in the introduction is step (3), making a database. It is to this end that the second argument of the "`ru:back`" function has been filled with a semicolon-separated list of feature values. When the queries have been run on the input texts within the CorpusStudio program, an *xml* file that contains all the important information on the result sentences is created, but this is not yet the database. This correct part of this result file can be transformed into an *xml* database by pressing a button within CorpusStudio, labelled "**create result database**".

Figure 1 provides a screenshot of the relevant part of CorpusStudio, called the "ConstructorEditor". This editor contains the queries that are to be processed for the currently loaded corpus research project, and it defines their hierarchical order. The CorpusStudio manual describes the process involved in generating a database from the results of a query line in more detail [5].

[4] This feature is unnecessary for the current example, where we only look at present-tense progressive inversion.

[5] The Xquery functions starting with the "`ru:`" prefix are all listed in the CorpusStudio manual. These functions have been hard-coded in CorpusStudio and approach the *xml* documents through the Microsoft *xml* library; the program makes use of the Saxon Xquery dll, which, in turn, allows host-programs to provide additional Xquery functions through a namespace declaration that points to the executable itself. The `ru:RefState` function is described more fully in [8].

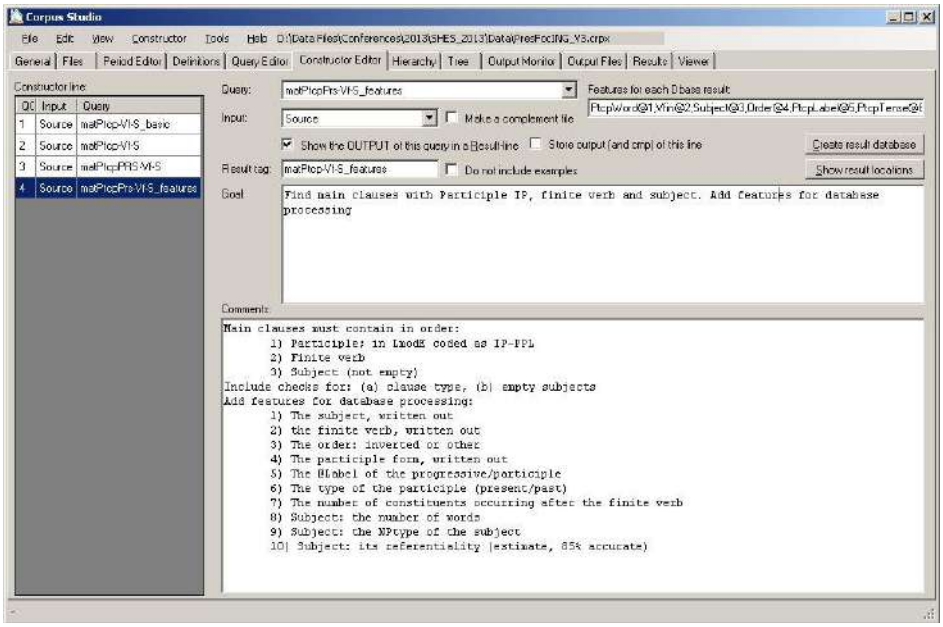


Figure 1 Creating an *xml* database in the Constructor Editor of CorpusStudio

6 Editing examples in the database

Steps (1) to (3) involved in working through a linguistics example, as described in the introduction, have been taken, and everything is ready for step (4), manually editing and inspecting the database. Loading the database in Cesax results in the following display.

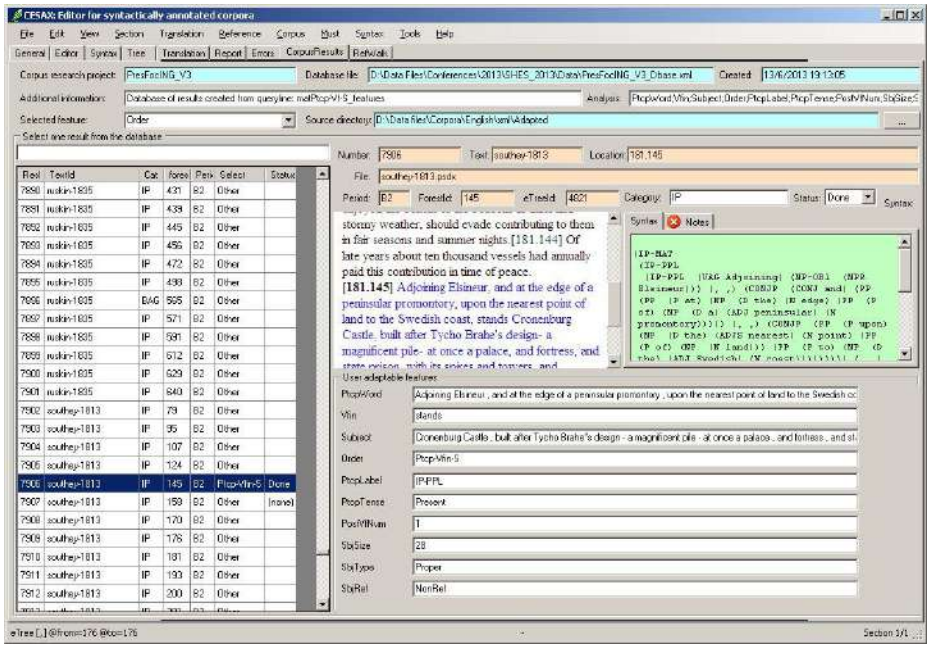


Figure 2 Loading an *xml* database in Cesax

The Cesax program has originally been created to facilitate coreference resolution and referential state processing, but it has been extended with several more functions, one of which is the editing of databases. Once a database has been loaded, editing options become available on the "CorpusResults" tab page [6].

1) **Delete**. Individual records can be deleted, but it is also possible to keep the records that are available, and indicate their status as "Ignore".
2) **Add**. If important sentences have not been captured by the database construction query, it is better to adapt the query in such a way that all sentences are added.
3) **Editing**. Feature values can be edited in the textboxes available for each record.
4) **Notes**. The "Notes" window allows adding remarks to individual records
5) **Status**. The status of each record can be set in order to keep track of progress.
6) **Bulk-changes**. Two different methods are provided to provide a search and replace feature. The most extensive option uses Xpath to find its way through the results in the database *xml* file, but it uses a user-friendly interface.

The database results can be re-ordered on the basis of any of the columns, and one column can be filled with one of the user-supplied features. It is also possible to *filter* the database without actually changing its content. These kinds of features make life easier for the annotator, especially when databases are large (the databases with results I have encountered typically exceed 10.000 sentences).

The syntax and local context of each record in the database are immediately visible in the "CorpusResults" tab page, but it may, at times, be necessary to look at the sentence that has been found in the larger context of the original text. Cesax allows this: double clicking the entry in the results list opens the corpus file on the corresponding place and shows it in the "Editor" tab page. Should it be necessary to take a different look at the syntax of this particular example, then clicking the "Tree" tab page results in displaying the selected sentence in a syntactic tree.

7 Preparation for statistics

Step (5) in the process described in the introduction involved preparing the database results for statistical processing. Cesax contains several commands to suit the needs of the user. Preparation for SPSS processing, for instance, involves the following steps:

1) Construct a table with the 'original' values of the features; the values as they are visible in the CorpusResults tab page;

2) Construct a tab-separated text file where the 'original' feature values are replaced by numerical values (an additional table with the 'index' to these values is supplied separately);
3) Construct a separate .sps file (an SPSS 'syntax' file).

Work with SPSS can be conducted by transferring the second (numerical value) table to SPSS, and processing the .sps file with the feature values. An SPSS user will, in addition to this, also need to specify which features are to be excluded from statistic, which are the independent variables, and which one is the dependent variable.

Work with memory-based language programs like 'TiMBL' is also supported [4]. Cesax allows preparing a training and test file with the necessary features for further processing by TiMBL.

Since the purpose of this paper is to show *how* data gained through corpus searches can be prepared for statistical processing, no attempt will be made to figure out which of the independent variables play a role in determining whether progressive inversion occurs or not.

8 Querying a database

Once a database has been manually edited, as described in section 6, a user will probably not want to go back to adjusting the original corpus query (section 3) in order to make a new version of a database (e.g. one that contains a selected subset, or one with adjusted feature values). This may, due to the cyclic process of research in general, not always be circumvented, but the CorpusStudio-Cesax combination does allow for one way out. If a user wants to make an adapted database that (a) uses a subset of the features available in the original one, or (b) that has records filtered out by additional criteria, or (c) that uses additional features that can be calculated on the basis of the existing ones, then this can be achieved by writing a query with the database as input. The CorpusStudio manual contains information on how to do this.

Returning now to the linguistic task that has been undertaken as an example, I would not like to withhold the outcome to the interested reader. The manually inspected corpus database yields a total of twelve examples of the progressive inversion (against a total of 5-6 million words), and the first clear one is found in early Modern English (1500-1700).

(5) a. and vpon the ryght hande goynge from Rama to Jherusalem, about .xx. myle from Rama, <u>is</u> **the castell of Emaus**. [chaplain-e1-p2:289]

The example in (5) has the finite verb *is* preceded by a participle clause that is headed by the present participle *going*. It clearly serves to introduce a new 'participant' in the narrative, namely the castle of 'Emaus'.

9 Discussion

This paper has shown a new, windows-based approach to research into variation and change of syntactic constructions. The new approach is centered around the programs CorpusStudio and Cesax, and makes heavy use

of *xml*, *xpath* and *xquery*, which have become standard public-domain conventions.

Just as CorpusSearch [9], tgrep [10], TigerSearch [3] and similar query programs do, CorpusStudio allows for the definition of queries that select sentences from syntactically parsed texts on the basis of user-definable criteria. Just as the Alpino project [2, 13] does, CorpusStudio makes use of the Xquery language with all its advantages in terms of user-extensibility, recursive functions and independent W3C development. Different from its competitors, however, CorpusStudio allows for combining multiple queries into a *corpus research project* that is kept in one place, which facilitates experiment replicability. Essential for the creation of a database with examples is CorpusStudio's capability to provide the examples that are found with pre-calculated feature values. This capability surpasses, for instance, CorpusSearch's "coding" functionality; first in the area of user-friendliness, and second in terms of complexity. Pre-calculating feature values in CorpusStudio is "advanced", since it can make use of the Xquery functionality of user-definable functions, and it can make use of the Xquery functions that have been hard-wired into CorpusStudio.

Since databases that have been made with CorpusStudio contain features that can have text values, editing such databases becomes a doable task. When database entries are also supplied with notes, the data become a valuable treasure, that allow back-tracking annotation choices. The facility to jump to the location in the text associated with a database entry allows for speedy inspection of the larger context, and it opens the way to a tree-view of the selected sentence's syntax.

Cesax allows simple transformation of a database into a format that can be used by statistical programs such as "R" and "SPSS", as well as by memory-based learning programs such as "TiMBL".

I suggest that future developments of Corpus databases based on treebanks involve web interfaces instead of dedicated programs (which tend to be OS-dependant), but I leave that challenge to the experts.

10 References

[1] Boag, Scott, Chamberlin, Don, Fernández, Mary F., Florescu, Daniela, Robie, Jonathan, and Siméon, Jérôme (2010) XQuery 1.0: An XML Query Language (Second Edition) W3C Recommendation.

[2] Bouma, Gosse (2008) XML information extraction with Xquery: processing wikipedia and Alpino trees. In Editor (ed.)^(eds.): 'Book XML information extraction with Xquery: processing wikipedia and Alpino trees' (Information science, university of Groningen, edn.), pp.

[3] Brants, Sabine, Dipper, Stefanie, Eisenberg, Peter, Hansen-Schirra, Silvia, König, Esther, Lezius, Wolfgang, Rohrer, Christian, Smith, George, and Uszkoreit, Hans (2004) TIGER: Linguistic Interpretation of a German Corpus. *Research on Language and Computation* 2, (4), 597-620.

[4] Daelemans, Walter, and Bosch, Antal van den (2005) Memory-based language processing (Cambridge University Press, 2005)

[5] Komen, Erwin R. (2009) Corpus Studio manual. Nijmegen: Radboud University Nijmegen.
[6] Komen, Erwin R. (2011) Cesax: coreference editor for syntactically annotated XML corpora. Reference manual. Nijmegen, Netherlands: Radboud University Nijmegen.
[7] Komen, Erwin R. (2012) Coreferenced corpora for information structure research. In Tyrkkö, Jukka, Kilpiö, Matti, Nevalainen, Terttu, and Rissanen, Matti (eds.) *Outposts of Historical Corpus Linguistics: From the Helsinki Corpus to a Proliferation of Resources. (Studies in Variation, Contacts and Change in English 10)*. Helsinki, Finland: Research Unit for Variation, Contacts, and Change in English.
[8] Komen, Erwin R. (2013) Predicting referential states using enriched texts. In Editor (ed.)^(eds.): 'Book Predicting referential states using enriched texts' (edn.), pp.
[9] Randall, Beth, Taylor, Ann, and Kroch, Anthony (2005) http://corpussearch.sourceforge.net, accessed 2/Jun/2009
[10] Rohde, Douglas L. T. (2005) TGrep2 user manual
[11] Sperberg-McQueen, C.M., and Burnard, Lou (2009) TEI P5: Guidelines for Electronic Text Encoding and Interchange (TEI Consortium, 2009)
[12] Ward, Gregory L., and Birner, Betty J. (1992) VP inversion and aspect in written texts. In Stein, Dieter (ed.) *Co-operating with written texts : the pragmatics and comprehension of written texts*, pp. 575-588. Berlin; New York: Mouton de Gruyter.
[13] Yao, Xuchen, and Bouma, Gosse (2010) Mining Discourse Treebanks with XQuery. In Editor (ed.)^(eds.): 'Book Mining Discourse Treebanks with XQuery' (edn.), pp. 245-256

Annotation and Disambiguation of English Compound Units in the English DeepBank

Valia Kordoni

Department of English and American Studies
Humboldt-Universität zu Berlin
E-mail: evangelia.kordoni@anglistik.hu-berlin.de

Abstract

The aim of this paper is twofold. We focus, on the one hand, on the task of dynamically annotating English compound nouns, and on the other hand we propose disambiguation methods and techniques which facilitate the annotation task. These annotations are very rich linguistically, since apart from syntax they also incorporate semantics, which does not only ensure that the treebank is guaranteed to be a truly sharable, re-usable and multi-functional linguistic resource, but also calls for the necessity of a better disambiguation of the internal (syntactic) structure of larger units of words, such as compound units, since this has an impact on the representation of their meaning, which is of utmost interest if the linguistic annotation of a given corpus is to be further understood as the practice of adding interpretative linguistic information of the highest quality in order to give "added value" to the corpus.

1 Introduction

Disambiguating compounds is a challenging task for several reasons. The first challenge lies in the fact that the formation of compounds is highly productive. This is not only true for English, but for most languages in which compounds are found. Secondly, both the annotation and the disambiguation of compounds is particularly tricky in English, for there are no syntactic and hardly any morphological cues indicating the relation between the nouns: as has very often to date been proposed in the relevant literature, the nouns are connected by an implicit semantic relation. Being a true Natural Language Processing task, the third difficulty in compound noun annotation and disambiguation lies in ambiguity. One could say that compound nouns are double ambiguous: a compound may have more than one possible implicit relation. Therefore, the interpretation of the compound may also depend on context and pragmatic factors. The last main challenge lies in the fact that, even though finite sets of possible relations have been proposed (by among

others [12], [25]), there is no agreement on the number and nature of semantic relations that may be found in compounds. Since [6], it is generally assumed that theoretically, the number of possible semantic relations is infinite.

The annotation and disambiguation of the English compound units we focus on here are part of the English DeepBank, an on-going project whose aim is to produce rich syntactic and semantic annotations for the 25 Wall Street Journal (WSJ) sections included in the Penn Treebank (PTB: [14]). The annotations are for the most part produced by manual disambiguation of parses licensed by the English Resource Grammar (ERG: [7]), which is a hand-written, broad-coverage grammar for English in the framework of Head-driven Phrase Structure Grammar (HPSG: [18]).

The aim of the DeepBank project [8], has been to overcome some of the limitations and shortcomings which are inherent in manual corpus annotation efforts, such as the German Negra/Tiger Treebank ([1]), the Prague Dependency Treebank ([9]), and the TüBa-D/Z.[1] All of these have stimulated research in various sub-fields of computational linguistics where corpus-based empirical methods are used, but at a high cost of development and with limits on the level of detail in the syntactic and semantic annotations that can be consistently sustained. The central difference in the DeepBank approach is to adopt the *dynamic* treebanking methodology of Redwoods [17], which uses a grammar to produce full candidate analyses, and has human annotators disambiguate to identify and record the correct analyses, with the disambiguation choices recorded at the granularity of constituent words and phrases. This localized disambiguation enables the treebank annotations to be repeatedly refined by making corrections and improvements to the grammar, with the changes then projected throughout the treebank by reparsing the corpus and re-applying the disambiguation choices, with a relatively small number of new disambiguation choices left for manual disambiguation.

2 Annotation of Compound Units

The annotation of DeepBank as a whole, and thus also of the compounds in this text collection, is organised into iterations of parsing, treebanking, error analysis, and gramma/treebank update cycles.

2.1 Parsing

Each section of the WSJ corpus is first parsed with the PET unification-based parser [3] using the ERG, with lexical entries for unknown words added on the fly based on a conventional part-of-speech tagger, TNT [2]. Analyses are ranked using a maximum-entropy model built using the `TADM` [13] package, originally trained on out-of-domain treebanked data, and later improved in accuracy for this task by including a portion of the DeepBank itself for training data. A maximum of 500

[1] http://www.sfs.nphil.uni-tuebingen.de/en_tuebadz.shtml

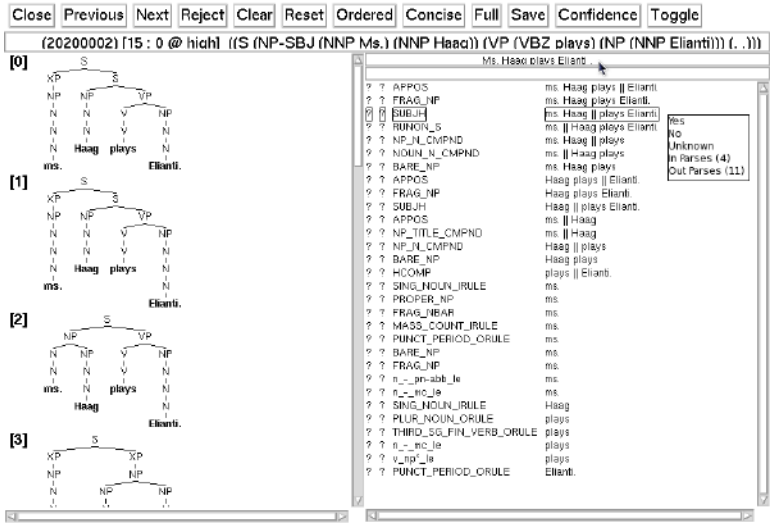


Figure 1: Treebanking Interface with an example sentence, candidate readings, discriminants and the MRS. The top row of the interface is occupied by a list of functional buttons, followed by a line indicating the sentence ID, number of remaining readings, number of eliminated readings, annotator confidence level, and the original PTB bracket annotation. The left part displays the candidate readings, and their corresponding IDs (ranked by the disambiguation model). The right part lists all the discriminants among the remaining readings. The lower part shows the MRS of one candicate reading.

highest-ranking analyses are recorded for each sentence, with this limit motivated both by practical constraints on data storage costs for each parse forest and by the processing capacity of the `[incr tsdb()]` treebanking tool [16]. The existing parse-ranking model has proven to be accurate enough to ensure that the desired analysis is almost always in these top 500 readings if it is licensed by the grammar at all. For each analysis in each parse forest, we record the exact derivation tree, which identifies the specific lexical entries and the lexical and syntactic rules applied to license that analysis, comprising a complete 'recipe' sufficient to reconstruct the full feature structure given the relevant version of the grammar. This approach enables relatively efficient storage of each parse forest without any loss of detail.

2.2 Treebanking

For each sentence of the corpus, the parsing results are then manually disambiguated by the human annotators, using the `[incr tsdb()]` treebanking tool which presents the annotator with a set of binary decisions, called *discriminants*, on the inclusion or exclusion of candidate lexical or phrasal elements for the desired analysis. (cf., Figure 1).

This discriminant-based approach of [5] enables rapid reduction of the parse

forest to either the single desired analysis, or to rejection of the whole forest for sentences where the grammar has failed to propose a viable analysis.[2] On average, given n candidate trees, $\log_2 n$ decisions are needed in order to fully disambiguate the parse forest for a sentence. Given that we set a limit of 500 candidate readings per sentence, full disambiguation of a newly parsed sentence averages no more than 9 decisions, which enables a careful annotator to sustain a treebanking rate of 30 to 50 sentences per hour on the first pass through the corpus.

2.3 Error analysis

During the course of this annotation effort, several annotators have been trained and assigned to carry out the initial treebanking of portions of the WSJ corpus, with most sections singly annotated. On successive passes through the treebank, two types of errors are identified and dealt with: mistakes or inconsistencies of annotation, and shortcomings of the grammar such that the desired analysis for a given sentence was not yet available in the parse forest. Errors in annotation include mistakes in constituent boundaries, in lexical choice such as verb valency or even basic part of speech, and in phrasal structures such as the level of attachment of modifiers or the grouping of conjuncts in a coordinated phrase. Our calculation of the inter-annotator agreeemnt using the Cohen's KAPPA[4] on the constituents of the derivation trees after the initial round of treebanking shows a moderate agreement level at $\kappa = 0.6$. Such disagreeements are identified for correction both by systematic review of the recorded 'correct' trees section by section, and by searching through the treebank for specific identifiers of constructions or lexical entries known to be relatively rare in the WSJ, such as the rules admitting questions or imperative clauses.

Shortcomings of the grammar are identified by examining sentences for which annotators did not record a correct analysis, either because no analysis was assigned, or because all of the top 500 candidate analyses were flawed. Some of the sources of error emerge quickly from even cursory analysis, such as the initial absence of a correct treatment in the ERG for measure phrases used as verbal modifiers, which are frequent in the WSJ corpus, as in *the index rose 20 points* or *the market fell 14%*. Other types of errors required more detailed analysis, such as missing lexical entries for some nouns taking verbal complements, as in *the news that Smith was hired* or *the temptation to spend the money*. These fine-grained lexical entries are not correctly predicted on the fly using the part-of-speech tagger, and hence must be added to the 35,000-entry manually supplied lexicon in the ERG.

[2]For some sentences, an annotator may be unsure about the correctness of the best available analysis, in which case the analysis can still be recorded in the treebank, but with a lower 'confidence' score assigned, so the annotation can be reviewed in a later cycle of updates.

2.4 Grammar & Treebank Update

While grammar development proceeds independent of the initial treebank annotation process, we have periodically incorporated improvements to the grammar into the treebank annotation cycle. When a grammar update is incorporated, the treebank also gets updated accordingly by (i) parsing anew all of the sentences in the corpus using the new grammar; (ii) re-applying the recorded annotation decisions; and (iii) annotating those sentences which are not fully disambiguated after step ii, either because new ambiguity was introduced by the grammar changes, or because a sentence which previously failed to parse now does. The extra manual annotation effort in treebank update is relatively small when compared to the first round of annotation, typically requiring one or two additional decisions for some 5–10% of the previously recorded correct analyses, and new annotation for previously rejected items, which were another 15% of the total in the second round, and much less in successive rounds. Hence these later rounds of updating the treebank proceed more quickly than the initial round of annotation.

Correcting errors of both classes based on analysis of the first pass through DeepBank annotation has resulted in a significant improvement in coverage and accuracy for the ERG over the WSJ corpus. Raw coverage has risen by some 10% from the first pass and the 'survival' rate of successfully treebanked sentences has risen even more dramatically to more than 80% of all sentences in the first 16 sections of the WSJ that have now gone through two rounds of grammar/treebank updates.

3 Examples of Compound Units in DeepBank

Being a collection of financial articles, the WSJ may not represent the English language in its most typical daily usage, but it is not in short of interesting linguistic phenomena. Having an average sentence length of over 20 words, loaded with tons of jargons in the financial domain, the corpus puts many natural language processing components (POS taggers, chunkers, NE recognizers, parsers) to the ultimate test. On the other hand, rich phenomena included in the corpus make it also interesting to test deep linguistic processing techniques. One particularly frequent and puzzling phenomenon in the corpus is the vast amount of compound nouns whose syntactic and semantic analyses are potentially ambiguous. Being symbolic systems, deep grammars like the ERG will not always disambiguate all the possibilities. For example, for the compound *"luxury auto maker"*, the ERG will assign both left-branching and right branching analyses (as shown in Figure 2), using the very unrestricted compounding rule NOUN-N-CMPND.

In some cases such branching decisions seem arbitrary and are defensible either way, but there are instances where a distinction should be made clearly. Consider the following two sentences from the WSJ section 3 of the PTB:

- *A form of asbestos once used to make **Kent cigarette filters** has caused a*

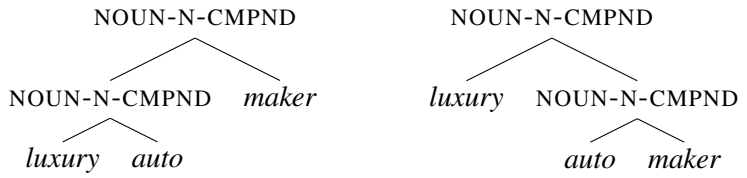


Figure 2: Two alternative analyses from the ERG

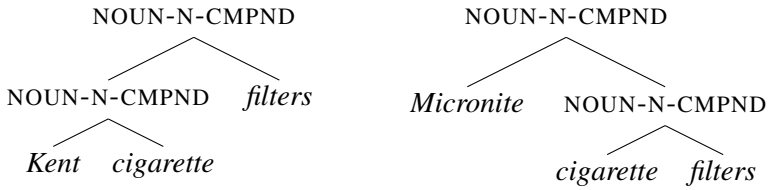


Figure 3: Similar noun compounds with different branching preferences

high percentage of cancer deaths among a group of workers exposed to it more than 30 years ago, researchers reported.

- *Lorillard Inc., the unit of New York-based Loews Corp. that makes Kent cigarettes , stopped using crocidolite in its* ***Micronite cigarette filters*** *in 1956.*

In some cases, such branching preferences can be easily accounted for, if part of the compound is a multiword named entity, as in "Fortune 500 executives" and "auto maker Mazda Motor Corp.", where the words from the named entity should be grouped together.

More challenging cases come from the financial domain specific terminologies. While the majority of such terminologies conform to the largely right-branching structures of English, there are cases where left-branching structures may not be excluded in the analysis of the given compounds.

- *Nevertheless , said Brenda Malizia Negus, editor of Money Fund Report , yields " may blip up again before they blip down " because of recent rises in* ***short-term interest rates****.*
- *Newsweek said it will introduce the* ***Circulation Credit Plan*** *, which awards space credits to advertisers on "renewal advertising."*

While varying branching preference can hopefully be recovered partially by a statistical disambiguation model trained on the increasing number of manually disambiguated compounds in the treebanking project, there are also problems which need special treatment in the design of features for the disambiguation model. For instance, in a compound construction containing a deverbal noun, the predicate-argument relation from the deverbal noun to the other noun in the compound is

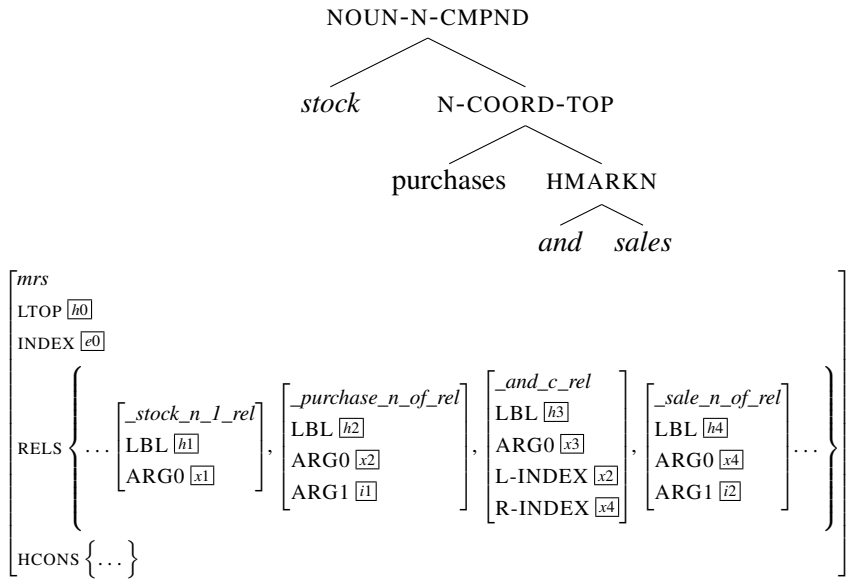


Figure 4: Missing semantic relation within a compound

left underspecified by the grammar, for the relation can be either an argument or a modifier. Consider the compound *"stock purchase and sales"*. A valid syntactic analysis (as shown in Figure 4) leaves an unbound semantic relation.

Ideally, in this example the semantic variables *i1* and *i2* should be both bound to *x1*. But resolving such an ambiguity within the grammar involves the risk of wrongly assigning the semantic roles in cases where, say, the first noun is serving as modifier instead of argument of the deverbal noun. The current disambiguation model does not recover such a kind of underspecified semantic information, as the model is trained exclusively on disambiguated treebanked data with underspecified semantics unchanged. Furthermore, such disambiguation requires a big number of bi-lexical preferences, in order, for instance, for the distinction between arguments and modifiers to be drawn clearly.

4 Disambiguation of Compound Nouns

Due to the lack of constraints on compound nouns in the ERG, the grammar tends to generate all possible internal structures to these NPs, leading to an combinatorial explosion to the number of candidate trees. Working with the DeepBank, we delay the decision on these internal structures of compounds until the other parts of the syntactic structures are disambiguated. Then the annotators go on to pick the preferred branching structures in line with the examples shown in the previous section.

The human annotators have been assisted with several disambiguation models

that help to rank the readings and treebanking decisions. The annotators have been warned to make use of this help with cautiousness. The inter-annotator agreements have been checked periodically to ensure the quality of the annotation.

In need to further facilitate and boost the performance of the parse disambiguation model used for the annotation of compounds in the DeepBank, we have also adopted the following strategies:

1. Almost in a preprocessing component manner, we have relied on the detection and evaluation methods for the automatic acquisition of Multiword Expressions (MWEs), thus also of compound nouns, for robust grammar engineering proposed in [24]. That is, we have first detected compound noun candidates for English on the basis of the distinct statistical properties of their component words, regardless of their type, comparing 3 statistical measures: mutual information (MI), χ^2 and permutation entropy (PE). Then we have validated the quality of such candidates against various corpora, investigating simultaneously the influence of the size and quality of different corpora, using the BNC and the Web search engines Google and Yahoo. At the end of this process, the eligible compound noun candidates have been introduced to the ERG-based parsing and treebanking procedure with the aim to also get validated by the human annotators before ultimately being used for the re-training of the parse disambiguation model.

2. That is, in a novel manner, we have incorporated the fine-grained treebanking decisions made by the human annotators as discriminative features for the automatic parse disambiguation of the compounds in the DeepBank. The advantage of such an approach and everyday treebanking practice is that use of human judgements is made. [11] show that annotators tend to start with the decisions with the most certainty, and delay the "hard" decisions as much as possible. As the decision process goes, many of the "hard" discriminants pertaining to compounds have received an inferred value from the certain decisions. This greedy approach has been shown to help guarantee high inter-annotator agreement. Concerning the statistical parse selection model, the discriminative nature of such treebanking decisions suggests that they are highly effective features, and if properly used, they contribute to an efficient disambiguation model.

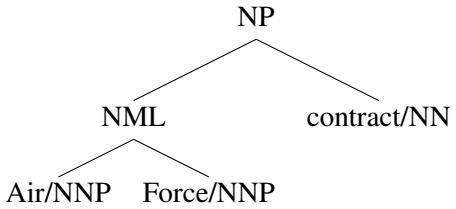
3. Use the annotated sections of the WSJ to retrain the parse disambiguation model and improve the syntactic bracketing prediction accuracy. The parse disambiguation model used here is that proposed in [20] and [19] which has been developed for use with so-called dynamic treebanking environments, like the Redwoods treebank [17]. In such a model, features such as local configurations (i.e., local sub-trees), grandparents, n-grams, etc., are extracted from all trees and used to build and (re-)train the model. Thus, as part of this procedure for our purposes, the eligible compound noun candidates have been introduced to the ERG-based parsing and treebanking procedure and

they have been validated through annotation by the human annotators before ultimately being used for the re-training of the inherent to the parser disambiguation model. The ultimate aim here has been to incorporate the fine-grained treebanking decisions made by the human annotators as discriminative features for the automatic parse disambiguation of the compounds in the DeepBank.

4. Use external large corpora to gather bi-lexical preference information as auxiliary features for the maximum-entropy based parse disambiguation model mentioned above. This is similar to the approach taken in [10] and [23], where pointwise mutual information association scores are used in order to measure the strength of selectional restrictions and their contribution to parse disambiguation. Because the association scores are estimated on the basis of a large corpus that is parsed by a parser and is aimed at getting improved through parse disambiguations, this technique may be described as a particular instance of self-training, which has been shown in the literature to serve as a successful variant of self-learning for parsing, as well. The idea that selection restrictions may be useful for parsing is not new. In our case at hand, i.e., the case of the disambiguation of compound nouns that we are interested in here, our approach and method is very much fine-tuned and targeted to the disambiguation of argument vs. modifier relations in the compound nouns.

5 Discussion and Outlook

In the work of [21], efforts to enrich the noun phrase annotations for the Penn Treebank have been reported. The extra binarization of the originally flat NP structures provides more information for the investigation of the internal structures of the compound nouns, although the enriched annotation adds very little information to the labels, and the semantic relations within the NPs are not explicitly revealed. More specifically, the work of [21] leaves the right-branching structures (which are the predominant cases for English) untouched, and just inserts labelled brackets around left-branching structures. Two types of new labels were assigned to these new internal nodes of the PTB NPs: NML or JJP, depending on whether each time the head of the NP is a noun or an adjective. Hence, in this analysis, for instance, the NP "Air Force contract" would receive the following structure:



As a consequence of such an annotation and treatment, *Air Force* as a unit is serving the function of the nominal modifier of *contract*.

Such enriched annotation enables one to investigate the bracketing preferences within the nominal phrases which was not available with the original PTB. By adapting the existing parsing models to use the enriched annotation, one can expect a fine-grained parsing result. Furthermore, this allows one to explore the treatment of NP in linguistically deep frameworks (see [22] for an example of such study in the framework of Combinatory Categorial Grammar (CCG)).

In DeepBank, the aim has always been to develop linguistic analyses independent from the PTB annotations. In the same spirit, we have decided not to incorporate the NP bracketing dataset from [21] directly during the annotation phase. On the other hand, as pointed out by the original PTB developers ([15]), asking annotators to directly annotate the internal structure of the base-NP significantly slows down the annotation process. We have made a similar observation in the DeepBank project. To help improve the annotation speed while maintaining quality, we have periodically updated the statistical models that re-rank the candidate trees and discriminants (binary decisions to made by human annotators) so that the manual decision making procedure has been made easier.

As an immediate next step in the research carried out for the dynamic annotation and disambiguation of English compound nouns in the DeepBank, we will compare the bracketing agreement with the NP dataset from [21].

References

[1] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, pages 24–41, 2002.

[2] Thorsten Brants. TnT - a statistical part-of-speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing (ANLP 2000)*, Seattle, USA, 2000.

[3] Ulrich Callmeier. Efficient parsing with large-scale unification grammars. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany, 2001.

[4] Jean Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22(2):249–254, 1996.

[5] David Carter. The treebanker: a tool for supervised training of parsed corpora. In *Proceedings of the ACL Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, pages 9–15, Madrid, Spain, 1997.

[6] Pamela Downing. On the creation and use of English compound nouns. *Language*, 53:4:810–842, 1977.

[7] Dan Flickinger. Accuracy vs. robustness in grammar engineering. In Emily M. Bender and Jennifer E. Arnold, editors, *Language from a Cognitive*

Perspective: Grammar, Usage, and Processing, pages 31–50. Stanford: CSLI Publications, 2011.

[8] Daniel Flickinger, Yi Zhang, and Valia Kordoni. Deepbank: A dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories. International Workshop on Treebanks and Linguistic Theories (TLT-11), 11th, November 30 - December 1, Lisbon, Portugal*, pages 85–96. EdiÃğÃţes Colibri, Lisbon, 2012.

[9] Jan Hajič, Alena Böhmová, Eva Hajičová, and Barbora Vidová-Hladká. The Prague Dependency Treebank: A Three-Level Annotation Scenario. In A. Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pages 103–127. Amsterdam:Kluwer, 2000.

[10] Mark Johnson and Stefan Riezler. Exploiting auxiliary distributions in stochastic unification-based grammars. In *Proceedings of the 1st NAACL conference*, pages 154–161, Seattle, USA, 2000.

[11] Valia Kordoni and Yi Zhang. Annotating wall street journal texts using a hand-crafted deep linguistic grammar. In *Proceedings of The Third Linguistic Annotation Workshop (LAW III)*, Singapore, 2009.

[12] Judith Levi. *The Syntax and Semantics of Complex Nominals*. Academic Press, INC, New York, 1978.

[13] Robert Malouf, John Carroll, and Ann Copestake. Efficient feature structure operations without compilation. *Natural Language Engineering*, 6:29–46, 2000.

[14] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[15] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[16] Stephan Oepen. [incr tsdb()] — competence and performance laboratory. User manual. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany, 2001.

[17] Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. The LinGO Redwoods treebank: motivation and preliminary applications. In *Proceedings of COLING 2002: The 17th International Conference on Computational Linguistics: Project Notes*, Taipei, Taiwan, 2002.

[18] Carl J. Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, USA, 1994.

[19] Kristina Toutanova, Christoper D. Manning, Dan Flickinger, and Stephan Oepen. Stochastic HPSG parse selection using the Redwoods corpus. *Journal of Research on Language and Computation*, 3(1):83–105, 2005.

[20] Kristina Toutanova, Christoper D. Manning, Stuart M. Shieber, Dan Flickinger, and Stephan Oepen. Parse ranking for a rich HPSG grammar. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories (TLT 2002)*, pages 253–263, Sozopol, Bulgaria, 2002.

[21] David Vadas and James Curran. Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 240–247, Prague, Czech Republic, 2007.

[22] David Vadas and James R. Curran. Parsing noun phrase structure with CCG. In *Proceedings of ACL-08: HLT*, pages 335–343, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[23] Gertjan van Noord. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 1–10, Prague, Czech Republic, 2007.

[24] Aline Villavicencio, Valia Kordoni, Yi Zhang, Marco Idiart, and Carlos Ramisch. Validation and evaluation of automatically acquired multiword expressions for grammar engineering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 1034–1043, Prague, Czech, 2007.

[25] Beatrice Warren. *Semantic Patterns of Noun-Noun Compounds*. Acta Universitatis Gothoburgensis, Göteborg, 1978.

Polish LFG treebank on a shoestring

Katarzyna Krasnowska[1] Witold Kieraś[1,2]
k.krasnowska@phd.ipipan.waw.pl w.kieras@uw.edu.pl

[1]Institute of Computer Science, Polish Academy of Sciences
[2]Institute of Polish Language, University of Warsaw

Abstract

In the paper we present a method of partial disambiguation of an LFG parsebank produced by the Polish LFG grammar POLFIE. The method is based on the grammatical information retrieved from Składnica treebank consisting of the same set of sentences. As a result we obtain a parsebank consisting of significantly smaller forests of LFG structures that can be fully disambiguated by a human annotator with much less time and effort then in the case of entirely manual disambiguation.

1 Introduction

In this paper, we report on preliminary results concerning a method of semi-automatic creation of an LFG treebank of Polish based on an already existing resource. The aim of our work is to prune LFG forests obtained by automatic parsing with no means of stochastic disambiguation. The idea is to restrict them to trees consistent with already existent constituency annotation based on another grammar formalism for the parsed sentences. In this way, we perform an automatic, partial disambiguation which can be later completed by a human annotator. After the automatic stage of pruning, the annotators will be presented with much smaller parse forests, which will allow for a significant decrease of amount of time and human effort needed to obtain an LFG treebank for Polish.

Our approach is therefore a convenient alternative to more expensive (in terms of time and human effort) ways of creating a treebank, involving fully manual syntactic annotation (e.g., the Prague Dependency Treebank, Hajič et al. 2000) or disambiguation of the entire output of a parser (as in the Polish constituency treebank, Woliński et al. 2011) and therefore requiring more work from the annotators.

The strategy we use, i.e., making extensive use of an existing resource in order to obtain a new one, is in line with the "parasitic" approach adopted by the authors of POLFIE – the Polish LFG grammar (Patejuk and Przepiórkowski, 2012) – who used a Polish DCG[1] grammar as a base for their resource. Such an approach seems

[1]Definite Clause Grammar

promising especially in the case of languages with less developed NLP resources, such as Polish, since it allows for obtaining results in a relatively short time. As a side effect of this strategy we obtain two treebanks based on the same linguistic content which may allow future development of hybrid approaches to parsing.

This article describes our method and reports on its current results. Since both DCG treebank and POLFIE are still under development, the presented results are partial in a sense that first, the final version of the partially disambiguated LFG parsebank will consist of more sentences as the constituency treebank used as a starting point is about to grow, and second, the LFG grammar will be extended and, possibly, modified.

The paper is organised as follows. In section 2 the used resources, i.e., the constituency treebank of Polish sentences and the Polish LFG grammar, are briefly presented. Section 3 describes our method. Section 4 presents the current state of work.

2 Available resources

The Polish constituency treebank used in this work is Składnica[2] (Woliński et al., 2011). The treebank is being developed in a semi-automatic manner. Sets of candidate parse trees are generated by a parser and subsequently one tree is selected by human annotators. The tool used for parsing is Świgra (Woliński, 2004) – a parser for Polish based on a DCG-style grammar GFJP (Świdziński, 1992). The resource consists of about 8000 trees but is still under development. There is an ongoing work aiming at extending the treebank with more sentences and correcting the existing structures. Figure 1 shows an example tree from Składnica.

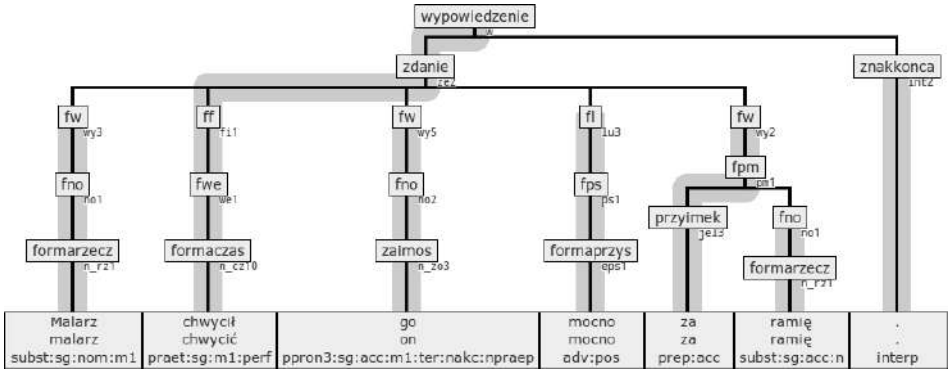


Figure 1: An example constituency tree from Składnica treebank. The highlighted edges lead to nodes marked as head elements.

[2] http://zil.ipipan.waw.pl/Składnica

Another resource is POLFIE[3] – a manually developed wide-coverage LFG[4] grammar for Polish (Patejuk and Przepiórkowski, 2012). Figure 2 shows an LFG representation of the sentence from Figure 1. The Polish LFG grammar was used to create a parsebank consisting of LFG structure forests for the sentences contained in Składnica. The next step is to transform the parsebank into a treebank by selecting the correct analysis from the forest.

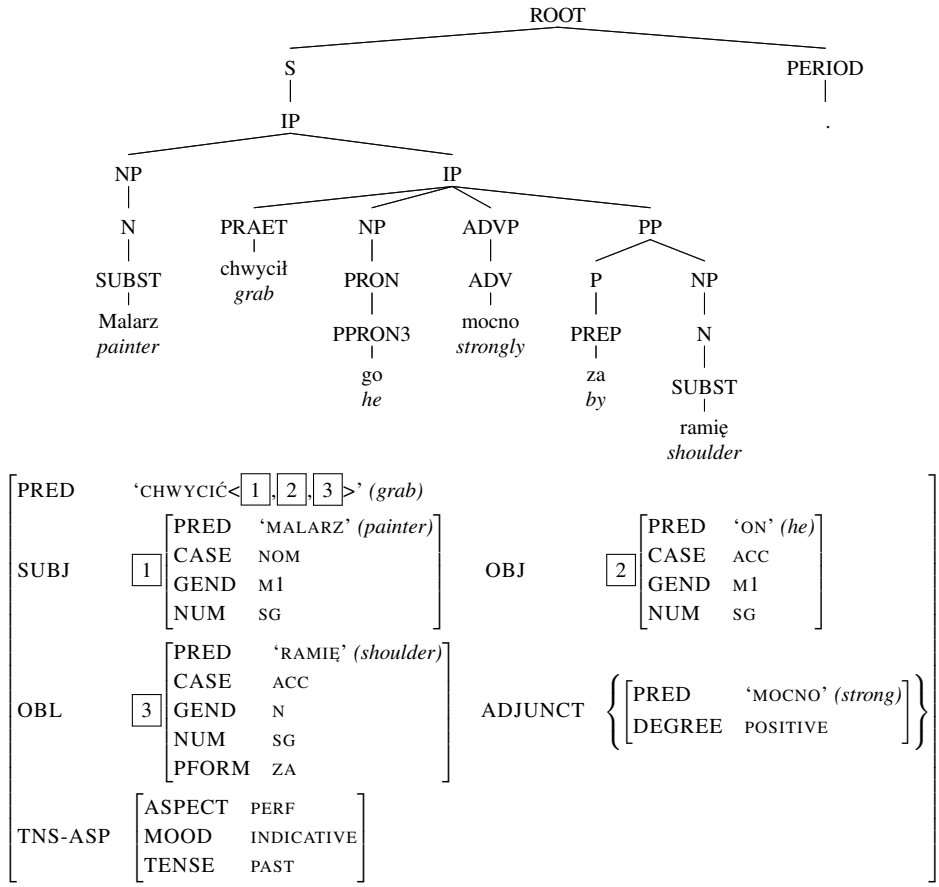


Figure 2: An example of LFG analysis: c-structure (top) and f-structure (bottom).

3 Method

The idea of our approach is to compare the hierarchy of predicates in LFG f-structures with the marking of heads of constituents in Składnica's trees. One may consider taking into account c-structures instead on the LFG side, given the fact that they are constituency trees and, as such, should be easier to compare with

[3] http://zil.ipipan.waw.pl/LFG

[4] Lexical Functional Grammar (Bresnan 2001, Dalrymple 2001)

other constituency trees. However, we found some important reasons for focusing on the functional level of sentence description.

First of all, it is a common view among LFG grammarians that the c-structure plays a secondary role in sentence analysis. It can be seen as an auxiliary structure for constructing the functional one. Another reason for the f-structure being more interesting is that it is expected to be much more language-independent, while the c-structure must be much more language-specific in order to deal, e.g., with different word order. Further, the f-structure is more suited to augmenting with semantic information, which makes it a better representation of a sentence in NLP applications dealing with semantic analysis.

Last but not least, focusing on the f-structure is a better decision from the technical point of view, as the grammar we are working with will be subject to changes in the course of its ongoing development. Although we cannot predict its future modifications, we expect that they will affect the constituency structure more than the functional one.

The general idea of our method is quite straightforward. We use head element marking present is Składnica's annotation to retrieve information about predicate hierarchy the LFG f-structure should be consistent with. By predicate hierarchy we mean the structure of functional relations between f-structures corresponding to individual predicates: which predicate's f-structure is the value of an attribute of another. The procedure is recursive. In each visited node of a tree, the procedure goes down the path to the leaf corresponding to its subtree's head element. The base form in that leaf will be the predicate of an f-structure. The subtrees attached to the path by non-head branches correspond to nested f-structures which are values of attributes of the current one, their head elements' base forms being the predicates. As an example, from the constituency tree shown in Figure 3, the following information would be extracted:

- the f-structure for 'STRACIĆ' (*lose*) has as values of its attributes the f-structures for 'MILION' (*million*) and 'OSZCZĘDNOŚĆ' (*saving*)
- the f-structure for 'MILION' has as the value of its attribute the f-structure for 'CZŁOWIEK' (*man*)
- the f-structure for 'OSZCZĘDNOŚĆ' has as the value of its attribute the f-structure for 'ŻYCIE' (*life*)
- the f-structure for 'ŻYCIE' has as the value of its attribute the f-structure for 'CAŁY' (*whole*)

This information can be seen as a set of constraints that must be fulfilled by an f-structure in order to be consistent with the given sentence's constituency tree.

The basic method described above does not take into account what kind of attribute a nested f-structure is the value of. It does not distinguish between f-structures with analogous predicate hierarchy, but differing in specific functional relations. For instance, continuing the previous example, the f-structure for 'MILLION' may be, among others, SUBJ, OBJ or ADJUNCT (optional modifier) of

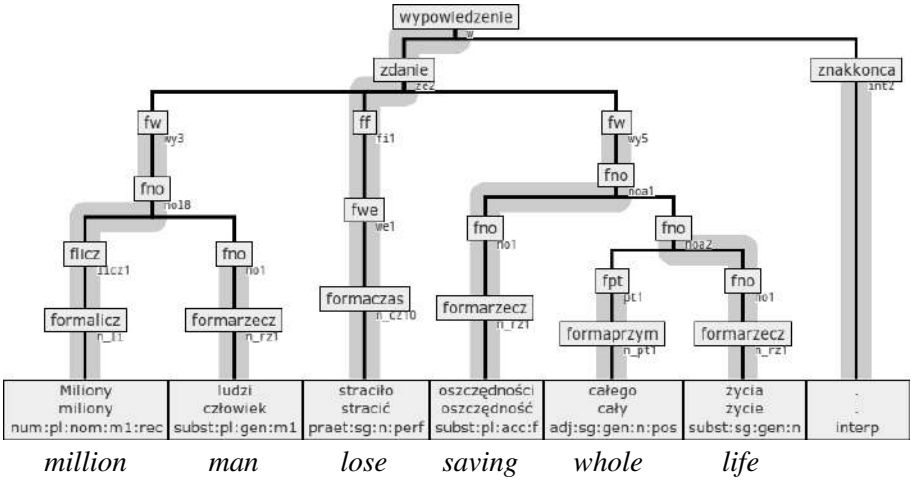


Figure 3: Składnica tree for sentence *Millions of people lost whole life's savings.*

the f-structure for 'LOSE'. All those cases are consistent with the information about hierarchy between 'MILLION' and 'LOSE' extracted from the tree.

Conveniently, Składnica contains some valence information allowing for making the extracted constraints more specific. The valence information is contained mainly at the level of the zdanie (sentence) node. Each of its children is labeled either ff (finite phrase, the sentence's main verb), fw (argument phrase) or fl (modifier phrase). Each fw node contains information about the valence position it fills. Valence annotation is also present in the case of participle verb forms and some nouns with argument requirements. This information is however oriented on surface syntax and therefore not isomorphic with LFG's grammatical functions.

Using this information and a mapping from valence positions to LFG attribute names, we can make the constraints more specific. In addition to the requirements on the predicates of each f-structure's attributes, we can state what should be the grammatical functions of those attributes. For the predicates for which the valence information is present, we split the attribute predicates in the constraint into two groups. One consists of the predicates extracted from argument phrases (argument predicates), and the other contains the rest (modifier predicates). When testing an f-structure for consistency with such a constraint, we check whether:

1. all argument predicates are matched in the f-structure, i.e., it has attributes with names and predicates specified in the constraint,
2. it has no other argument attributes[5],
3. its other attributes' predicates match the modifier predicates specified in the constraint.

[5]i.e., attributes that have a grammatical function, such as SUBJ, OBJ or OBL

Table 1 contains the mapping from valence positions marked in Składnica's argument phrase nodes to LFG attribute names. A valence position can be mapped to more than one LFG attribute. This is the case when, depending on a particular sentence, a phrase of the same syntactic type may appear as values of different attributes. Since the exact type of corresponding LFG attribute cannot be deduced directly from valence position marking, we assume that an f-structure satisfies a constraint if for each of its argument predicates, one of the attribute names specified for it is properly matched.

valence position	LFG attribute names
syntactic subject	SUBJ
nominal noun phrase	XCOMP-PRED, OBL-STR
genitive noun phrase	OBJ,OBL-GEN
dative noun phrase	OBJ, OBJ-TH
accusative noun phrase	OBJ, OBL-STR
instrumental noun phrase	OBL-INST, XCOMP-PRED, OBJ
nominal adjectival phrase	XCOMP-PRED
instrumental adjectival phrase	XCOMP-PRED
adverbial phrase	OBL (or modifier)
sentential phrase	SUBJ, COMP
infinitival phrase	SUBJ, XCOMP
prepositional noun phrase	OBL, OBL2, OBL3, OBL-AG
prepositional adjectival phrase	XCOMP-PRED

Table 1: Mapping from Składnica valence positions to POLFIE attribute names.

For example, in the tree from Figure 3, the only nodes with valence information are the children of the zdanie node. The marking is as follows: "Miliony ludzi" (*millions of people*) is marked as the syntactic subject and "oszczędności całego życia" (*whole life's savings*) as an accusative noun phrase. Thus, the constraint for 'STRACIĆ' will now state that its f-structure has two argument attributes: the f-structure for 'MILION' being a SUBJ and the f-structure for 'SAVING' as either OBJ or OBL-STR; and has no modifier attributes.

Since GFJP and POLFIE are not just two alternative formalisms for the same linguistic theory but rather two formalisms for two significantly different approaches to Polish syntax, a number of special cases needed to be taken into account to cover these differences. Some of such cases, for which the approach outlined above would not work directly, are listed below.

Subject. The first type of major difference are the cases in which there is no subject on the surface of a sentence. Such a situation is quite common in Polish, where an explicit subject is not required to be present in a sentence (so called "pro-drop"). For such sentences, there is simply no phrase marked as syntactic subject in their

constituency trees in Składnica. On the other hand, a subject is generally required in an LFG f-structure; in the case of no subject on the surface, a special f-structure with PRED 'PRO' takes its place to represent an implicit subject. We therefore had to take this fact into account and add a PRED 'PRO' subject to constraints for main verbs with no surface subject.

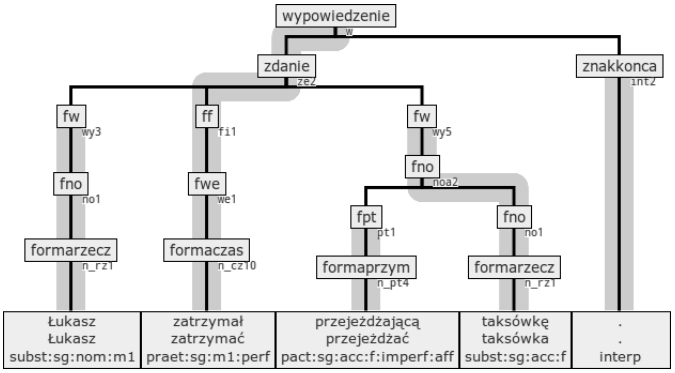
The handling of subject is also different in the case of participle verb forms. For example, in the sentence "Łukasz zatrzymał przejeżdżającą taksówkę." (*Łukasz stopped the passing cab.*; structures shown in Figure 4), Składnica's constituency representation contains no information about the subject of the verb form *passing*. On the other hand, in the corresponding LFG structure, the subject of 'PASS' is present and is identical with the object of 'STOP', which is 'CAB'. Since there is no direct way of identifying a participle verb's subject in a Składnica tree, we decided to accept any f-structure as its SUBJ in the LFG analysis. This could potentially lead to excessive growth in the number of incorrect structures accepted as consistent with Składnica, but guarantees that the correct ones are not rejected.

Another situation in which the subject will be different in Składnica and POLFIE is the case of coordination where a subject is shared between two conjoined sentences, as in *John ate and drank*. Although GFJP does handle some types of coordinated structures (e.g., coordinated noun phrases), it can only recognise *John ate* and *drank* (with an implicit subject) as two conjoined sentences. In POLFIE, the same sentence would receive two parses: one as above and another with 'EAT' and 'DRINK' sharing a common subject 'JOHN'. Since, without a semantic analysis, it is impossible to tell which one is correct, we cannot reject the second one based on the fact that *drunk* has no subject in Składnica. As a simple solution, we relax the constraints so that a requirement of a 'PRO' SUBJ is also satisfied by presence of any predicate in the place of SUBJ.

Prepositional phrases. In Składnica, all prepositional noun phrases have the same structure: they are headed by the preposition. In POLFIE, the way f-structure for a prepositional phrase is constructed depends on its role in a sentence. If a prepositional phrase is an argument (such as OBL), the main predicate of its f-structure is the main predicate of the noun phrase and the preposition is only marked as an atomic value attribute PFORM. If, however, a prepositional phrase is a modifier, then its main predicate is the preposition, and the f-structure for the noun phrase is its OBJ. This needed to be taken into account when comparing an f-structure with the constraints.

Coordination. Another important issue was coordination. In Składnica, a coordinated construction is represented as an exocentric phrase, with the conjunction being its head. In POLFIE, the representation of a coordinated phrase is an f-structure containing a set of f-structures for the conjuncts, and the conjunction is only marked as the COORD-FORM attribute (Patejuk and Przepiórkowski, 2012). For example, a constraint extracted from the Składnica tree shown in Figure 5

(1)



Łukasz stop pass cab

"Łukasz stopped the passing cab."

(2)

$$\begin{bmatrix} \text{PRED} & \text{'ZATRZYMAĆ}\langle [1], [2] \rangle\text{'} \textit{ (stop)} \\ \text{SUBJ} & [1]\begin{bmatrix}\text{PRED} & \text{'ŁUKASZ'}\end{bmatrix} \\ \text{OBJ} & [2]\begin{bmatrix}\text{PRED} & \text{'TAKSÓWKA' } \textit{(cab)} \\ \text{ADJUNCT} & \left\{\begin{bmatrix}\text{PRED} & \text{'PRZEJEŻDŻAĆ}\langle [2] \rangle\text{' } \textit{(pass)} \\ \text{SUBJ} & [2]\end{bmatrix}\right\}\end{bmatrix}\end{bmatrix}$$

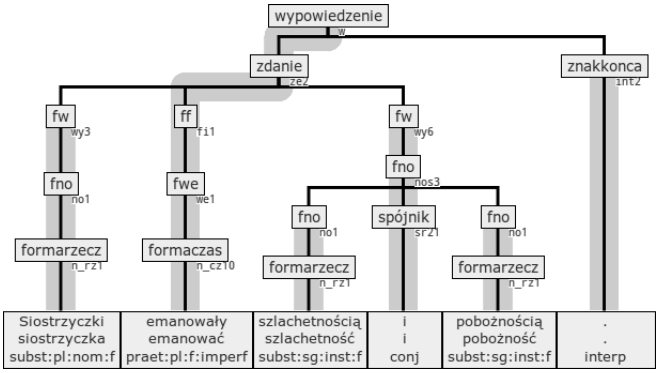
Figure 4: Example of structures for a sentence with active verb participle form: (1) in Składnica (2) in POLFIE (simplified f-structure only).

would state that the OBJ of 'EMANOWAĆ' has as its predicate the conjunction 'I', which is not possible in POLFIE. Instead, we ignore the conjunctions in the constituency trees and treat all the conjoined phrases as head elements. As a result, we obtain a constraint stating that 'EMANOWAĆ' has two OBJs: 'SZLACHETNOŚĆ' and 'POBOŻNOŚĆ'.

4 Experiments and results

There are currently 6513 sentences for which the currently available version of POLFIE produces non-empty parse forests. Out of those, 206 LFG analyses were manually disambiguated by an annotator in order to create a development data set. The implementation of our method was repeatedly tested against this set to ensure that our approach to pruning LFG parse forests is maximally consistent with the annotator's decisions (i.e., it does not reject the f-structures chosen as correct). Inspection of the development data showed that in the case of 19 sentences, it was impossible for our method to select the manually chosen parse based on Składnica due to different interpretations of the sentence (e.g., ambiguity of prepositional phrase attachment differently resolved by annotators working on Składnica and LFG parses). Thus, for 92% sentences from the development test, the LFG analysis selected by the annotator was among the ones pointed out as the most consistent

(1)



sister emanate nobility and piousness

"The sisters emanated nobility and piousness."

(2)

$$\begin{bmatrix} \text{PRED} & \text{'EMANOWAĆ<}\boxed{1}\text{,}\boxed{2}\text{>' } \textit{(emanate)} \\ \text{SUBJ} & \boxed{1}\begin{bmatrix}\text{PRED} & \text{'SIOSTRZYCZKA' } \textit{(sister)}\end{bmatrix} \\ \text{OBJ} & \boxed{2}\begin{bmatrix}\text{COORD-FORM I} \\ \left\{\begin{bmatrix}\text{PRED} & \text{'SZLACHETNOŚĆ' } \textit{(nobility)}\end{bmatrix}, \begin{bmatrix}\text{PRED} & \text{'POBOŻNOŚĆ' } \textit{(piousness)}\end{bmatrix}\right\}\end{bmatrix} \end{bmatrix}$$

Figure 5: Example of structures for a sentence with coordination: (1) in Składnica (2) in POLFIE (simplified f-structure only).

with Składnica.

The above observation suggests that instead of pruning the structures by preserving only those fully consistent with Składnica, another approach can be adopted. The LFG parses can be ranked according to the degree of accordance with the appropriate constituency tree, i.e., in the increasing order of unsatisfied constraints. In this way, we expect the correct LFG analysis to appear high in the ranking, but allow the annotator to look through all parses in case she finds none of the fully consistent ones appropriate. Another advantage of such approach is that it allows to smoothly handle the cases in which no f-structure satisfying all the constraints is found. This can be the case in situations where Świgra's grammar and POLFIE differ radically in their treatment of certain language phenomena. Such cases are difficult to predict due to both grammars still undergoing some changes, but will not lead to our method completely failing to produce a result for the "problematic" sentences.

After applying the method to the sentences with more than 1 and less than 10000 LFG analyses (there were 5730 such sentences[6]), we examined the number of parses with the minimal number of unsatisfied constraints (i.e., the highest ranked ones). Figure 6 shows two plots. The first one presents the number of such parses against the total number of parses for each sentence. The total number of parses ranged from 2 to 9720, with the median of 13 and mean of 158. The sec-

[6]737 sentences had only 1 parse and 46 had 10000 or more.

ond plot presents what percentage of the total number of parses is constituted by the highest ranked ones. The number of highest ranked parses varies from 2 to 200. The median and mean are 2 and 2.88 respectively, which is a satisfying result since it reduces the expected number of structures to be seen by the annotator significantly. The percentage of highest ranked parses decreases quickly with the increase of total number of parses, which shows that our method is most effective at pruning in the case of more complicated sentences which are the most difficult for the annotators.

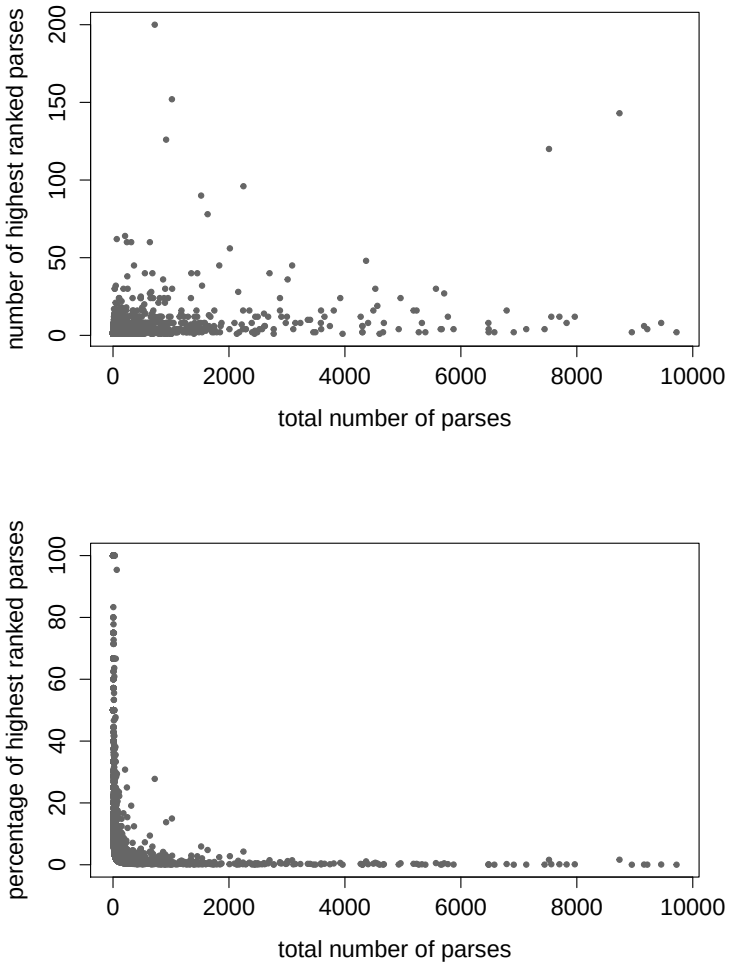


Figure 6: Number and percentage of highest ranked parses.

As can be seen in the plots, there are a few sentences for which the number of parses selected as the most consistent with Składnica is visibly larger than for sentences with similar total number of parses. This does not seem to be a seri-

ous problem given the small number of such cases (only for 1% of sentences there were more than 20 parses selected as the most consistent). Nevertheless, we examined some of those outlying sentences in order to determine the reasons for such a behaviour of our method.

One such type of sentence is one with multiple coordinated phrases. In its present form, our method does not check the internal structure of nested coordination, i.e., it does not make distinction between structures corresponding to different bracketings, such as *(A and B and C)*, *(A and (B and C))* or *((A and B) and C)*. It only checks whether all the conjuncts are present in the correct LFG attribute. As a result, when many phrases are coordinated, all possible nested structures for them are admitted as long as the internal structure of each coordinate is consistent with that in Składnica. This leads to an increase in the number of highest ranked parses. However, adding a mechanism for restricting the constraints for coordination would require deviating from the idea of looking only at the predicate hierarchy.

Another type are sentences with an initial of a name or surname. Since our method does not take into account any morphosyntactic information contained in the f-structure, it accepts structures with the initial being assigned any of 5 grammatical genders distinguished in POLFIE.

5 Related work

A work similar to ours in that it involved conversion of the same treebank to another formalism was performed by Wróblewska (2012). The cited paper reports on a fully automated, unambiguous procedure for creating dependency trees basing on Składnica's constituency parses. This approach however differs from ours since it does not make use of any dependency grammar, only of automatic conversion, and it does not assume any human disambiguation following the automatic stage, so the derived structures must be unique per sentence. Other experiments worth mentioning are those reported by Riezler et al. (2002) who also restricted LFG analyses to the ones consistent with the constituency trees. Unlike in our method, they annotated the sentences with WSJ bracketings before passing them to an LFG parser capable of using such information in order to reduce the number of produced parses.

6 Conclusion

The method presented above proved to be a useful tool for creating an LFG treebank from existing resource. Even though all used linguistic resources – Świgra, Składnica and POLFIE – are still being developed and are subject to change, it should be clear that our general method will require at most only minor improvements concerning handling special cases. As a result we can update the LFG treebank every time Składnica reaches next milestone.

Although the method proved to be successful in partial disambiguation of LFG parse forests, full automatisation of creating an LFG treebank on the basis of Składnica is not possible. GFJP and LFG are different formalisms based on different linguistic theories and some data required in POLFIE are simply not present in GFJP trees. Full disambiguation would require implementing some statistical methods.

The paper is cofounded by the European Union from resources of the European Social Fund. Project PO KL „Information technologies: Research and their interdisciplinary applications.

References

Joan Bresnan. *Lexical-Functional Syntax*. Blackwell Textbooks in Linguistics. Blackwell, 2001. ISBN 9780631209744. URL http://books.google.pl/books?id=vMxgevXoq_gC.

Mary Dalrymple. *Lexical Functional Grammar*. Syntax and Semantics. Academic Press, 2001. ISBN 9780126135343. URL http://books.google.co.uk/books?id=chEmQAAACAAJ.

Jan Hajič, Alena Böhmová, Eva Hajičová, and Barbora Vidová-Hladká. The Prague Dependency Treebank: A Three-Level Annotation Scenario. In A. Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, pages 103–127. Amsterdam:Kluwer, 2000.

Agnieszka Patejuk and Adam Przepiórkowski. Towards an LFG parser for Polish: An exercise in parasitic grammar development. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012*, pages 3849–3852, Istanbul, Turkey, 2012. ELRA.

Agnieszka Patejuk and Adam Przepiórkowski. A comprehensive analysis of constituent coordination for grammar engineering. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, Mumbai, 2012.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. Maxwell, III, and Mark Johnson. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 271–278, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. URL http://dx.doi.org/10.3115/1073083.1073129.

Marek Świdziński. *Gramatyka formalna języka polskiego*, volume 349 of *Rozprawy Uniwersytetu Warszawskiego*. Wydawnictwa Uniwersytetu Warszawskiego, Warszawa, 1992.

Marcin Woliński. *Komputerowa weryfikacja gramatyki Świdzińskiego*. PhD thesis, Institute of Computer Science, Polish Academy od Sciences, Warsaw, 2004.

Marcin Woliński, Katarzyna Głowińska, and Marek Świdziński. A preliminary version of Składnica—a treebank of Polish. In Zygmunt Vetulani, editor, *Proceedings of the 5th Language & Technology Conference*, pages 299–303, Poznań, 2011. ISBN 978-83-932640-1-8.

Alina Wróblewska. Polish dependency bank. *Linguistic Issues in Language Technology*, 7(1), 2012. URL http://elanguage.net/journals/index.php/lilt/article/view/2684.

Are All Commas Equal? Detecting Coordination in the Penn Treebank

Wolfgang Maier
University of Düsseldorf
Düsseldorf, Germany
maierw@hhu.de

Sandra Kübler
Indiana University
Bloomington, IN, USA
skuebler@indiana.edu

Abstract

Coordination has always been a difficult phenomenon, with regard to linguistic analysis, manual annotation, and automatic analysis. There is a considerable body of work on detecting coordination and on improving parsing for this phenomenon. However, most approaches are restricted to certain types of coordination, such as NP coordination or symmetrical coordination. We present the first approach to classifying punctuation signs into whether they function as separators between conjuncts in coordination or not. We show that by using information from a parser in combination with context information, we reach an F-score of 89.22 on the coordination case.

1 Introduction

The syntactic analysis and annotation of coordinated structures has generally been recognized as a difficult problem, in linguistics as well as in computational linguistics. Most linguistic frameworks still struggle with finding an account for coordination that is descriptively fully adequate [7]. In parsing approaches, coordinated structures constitute one of the main sources for errors [9]. Therefore, there are approaches in parsing that focus on improving parser performance specifically for this phenomenon [8, 9, 11, for example]. Others focus on detecting the scope of coordinations [6, 17]. However, most of these approaches focus on a closely defined subset of coordination types: either noun phrase coordination [9] or symmetric coordination of two conjuncts [13].

For English, one of the reasons for this focus lies in the annotation of the Penn Treebank (PTB) [15], which does not mark coordinated structures as such (for more details see section 3). For this reason, it is difficult to find certain coordinations, namely phrasal coordination without an overt conjunction, coordination on the clausal level, and coordinations with more than two conjuncts. However, there is an additional resource, which annotates all punctuation signs in the Penn

Treebank as to whether they function as conjunctions or not [14]. The current version of the annotation covers all sentences of the Penn Treebank release 3. Four annotators were involved.[1]

In this paper, we use this annotation in combination with the Penn Treebank to develop an automatic approach to detecting coordination and identifying its internal conjunct boundaries. More specifically, we interpret this task as a binary classification problem, in which a classifier decides whether a punctuation sign has a coordinating function, given its context, or not. If we can detect all punctuation signs, and combine them with the syntactic annotation, it is possible to determine the scope of the coordination, but also the number of conjuncts. In the current work, we focus on determining the types of information that are useful for the classification task: basic information such as part-of-speech (POS) tags, context words, or information about context punctuation; gold syntactic information, or syntactic information from a state-of-the-art parser.

The remainder of the paper is structured as follows: Section 2 describes work on parsing coordinations, section 3 gives our definition of coordination and describes the annotations in [14]. In section 4, we describe our experimental setup, in section 5 the results, and in section 6 our conclusions and future work.

2 Related Work

There are only a few approaches to explicitly identifying conjuncts. Ogren [17] concentrates on finding the full scope of coordinations with an overt conjunction. His method constructs simplified sentences out of coordinated sentences, each containing only one conjunct. He extracts possible left and right conjuncts and then evaluates their quality by a 4-gram language model. While Ogren is aware that there are coordinations with more than two conjuncts, he seems to group all conjuncts left of the conjunction into one maximal conjunct. Ogren's task is at the same time easier and more difficult than ours: He identifies only one left and one right conjunct of overt conjunctions, but then, his evaluation is rather strict in that the span of the conjuncts has to match the gold standard exactly.

Hara and Shimbo [6] work on the identification of the scopes of all coordinations with overt conjunctions in a sentence. Their input is a whole sentence, which they parse using a simple, manually written, overgenerating grammar for coordinations, in combination with a perceptron model to determine the optimal scope of potential conjuncts.

All other related works either aim at disambiguating specific subsets of coordinations or at improving parser models to better handle coordinated cases: Chantree et al. [2] approach cases of noun coordination involving one modifier, such as `old boots and shoes`. They use *SketchEngine* [12] to retrieve collocation statistics and distributional similarities. Based on this information, the candidates are ranked by heuristics. Dale and Mazur [4] tackle the problem of conjunction ambiguity in

[1] The resource will be made available soon via the Linguistic Data Consortium.

named entity recognition with a supervised approach. Bergsma and Yarowsky [1] use specialized classifiers based on bilingual, aligned data as well as on Google n-grams for the same task.

Integrating methods for improving coordination parsing has been performed on a wide range of languages: Hogan [9] integrates a specialized probability model for symmetrical coordinations into a parser for English. Guo et al. [5] use LFG-approximations to deal with long-distance dependencies in Chinese, including coordinations. Kübler et al. [13] work on German: They extract possible scopes for coordinations with two conjuncts and an overt conjunction and rerank them. Kawahara and Kurohashi [11] introduce a synchronized generation process into a generative dependency parser for Japanese. And Henestroza and Candito [8] use parse revisions to improve coordination, among other phenomena, in French.

3 Coordination Annotation

3.1 Coordination: The Linguistic Basis

Coordinations are complex syntactic structures that consist of two or more conjuncts. One or more of the conjuncts is often preceded by an (overt coordinating) conjunction, such as `and`, `or`, `neither...nor`, and `but`, see (1) for examples from the Penn Treebank. However, there are also cases of coordinations that lack coordinating conjunctions altogether, see (2).

(1) a. The total of 18 deaths from [malignant mesothelioma, lung cancer and asbestosis] was far higher than expected, the researchers said.
b. [Mr. Katzenstein certainly would have learned something, and it's even possible Mr. Morita would have too].
c. [The [Perch and Dolphin] fields are expected to start producing early next year, and the Seahorse and Tarwhine fields later next year].

(2) a. . . . a [humble, "uncomplaining, obedient] soul," . . .
b. Back in the chase car, we [drove around some more, got stuck in a ditch, enlisted the aid of a local farmer to get out the trailer hitch and pull us out of the ditch].

Coordinate structures typically exhibit syntactic parallelism in the sense that each conjunct belongs to the same lexical or phrasal category. However, coordinations of unlike categories are also possible, as in the example in (3), which conjoins the prepositional phrase `after the changes` and a clause `assuming no dramatic fluctuation in interest rates`. Typically, the conjuncts are syntactic constituents, but there are cases of non-constituent conjunctions, such as, e.g., in example (1-c), which involves an elliptical construction.

(3) He also said that [after the charges, and "assuming no dramatic fluctuation in interest rates], the company expects to achieve . . .

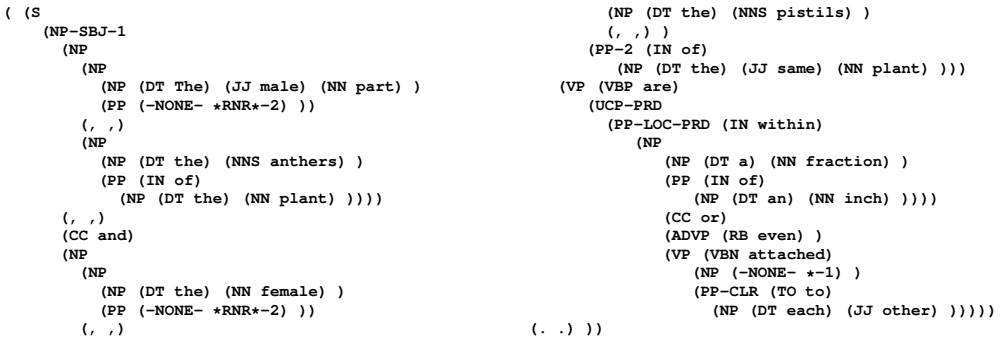


Figure 1: A PTB tree with a symmetrical and an asymmetrical coordination.

3.2 Coordination in the Penn Treebank

In the Penn Treebank [15], symmetrical coordination is generally annotated by grouping the conjuncts plus the conjunction under a node of the same type. The tree in figure 1 shows an example of a coordinated noun phrase (NP) consisting of two complex NPs, `The male part ...` and `the female ...`. Especially in cases without a conjunction, such constructions are difficult to distinguish from appositions, such as in (4-a), in which the two NPs `Elsevier N.V.` and `the Dutch publishing group` are also grouped under an NP. In some cases, even the presence of an overt conjunction is misleading when the conjunction introduces a different view of the same entity, as in (4-b). In this example, `EWDB` is another way of expressing the name of the company, and thus semantically an apposition.

(4) a. Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.
b. The transaction places the three executives squarely at the helm of a major agency with the rather unwieldy name of Eurocom WCRS Della Femina Ball Ltd., or EWDB.

Coordinations of unlike constituents are marked specifically, by a mother node `UCP`, as shown in figure 1 in the verb phrase (VP) where the PP `within a fraction of an inch` is coordinated with the VP `even attached to each other`. However, this is only true on the phrasal level; coordinations of unlike clauses are grouped under an S node without additional coordination marking.

3.3 Coordination Annotation: Making Coordination Explicit

In the version of the Penn Treebank annotated for coordination [14], almost all of these phenomena are annotated in the POS tag of the punctuation sign. The annotation focuses on the following punctuation signs: {, ; - – ...}. The only type of coordination not marked is coordination of clauses when there is no overt conjunct. Thus, examples (1-b) and (1-c) are marked for coordination, but (2-b) is

no. conjuncts	26	11	10	9	8	7	6	5	4
with CC	1	1	3	2	3	26	45	119	407
without CC	0	0	0	3	1	11	38	41	45

Table 1: An overview of the number of coordinations with many (≥ 4) conjuncts, separated into coordinations with and without overt conjunctions (CC).

not. If a punctuation sign serves more than one function and one is coordination, it is annotated as coordinating.

Additionally, the coordination annotation follows the syntactic annotation of the Penn Treebank as closely as possible. However, there are annotation errors in the treebank, such as the attachment of the comma after `plant` in figure 1, which should have been attached to the apposition `anthers of the plant`, not to the coordinated NP. In such cases, the coordination annotation deviates from the treebank annotation, and the comma after `plant` is annotated as non-coordinating.

Table 1 gives an overview of the number of coordinations with 4 or more conjuncts, with and without overt conjunctions over the whole treebank. These numbers show that we have a considerable number of coordinations with a high number of conjuncts, and we also have a non-negligible number of coordinations without an overt conjunction, i.e., cases where the conjuncts are separated by commas, semicolons, dots, or dashes. These frequencies show that it is not advisable to ignore either type of coordination, as has been the standard in previous work on parsing coordinations.

4 Experimental Setup

As mentioned before, we treat the problem of identifying conjunct boundaries which are marked by one of the punctuation signs in { , ; - – . . . } as binary classification task, in which a punctuation sign which marks / does not mark a conjunct boundary is classified as positive or negative instance, respectively, given the context in which it appears. For the training of the classifier, we use sections 02 to 21 of the WSJ part of the Penn Treebank. For testing, we use section 22. (We reserve section 23 for parsing results when integrating our annotated punctuation.) The test set contains 1 937 instances of non-coordination and 351 instances of coordinating punctuation. As usual for parsing, in all trees, we remove all traces and empty nodes and the corresponding co-indexation markers on non-terminals.

4.1 Classification

For our experiments we use *k*-nearest-neighbor classification as provided by TiMBL version 6.3 [3].[2] We describe the context of a single punctuation terminal by both

[2]We have also performed experiments with Support Vector Machines (SVM^{light}) [10], and a Maximum-Entropy classifier (MALLET version 2.07) [16]. However, since their performance was

word-level and syntactic features. Since the different punctuation signs have different distributions of usage in coordinations, we include the punctuation sign itself as a feature. The words and POS tags around a punctuation sign can be strong indicators for its function. Consider, for example, the words *and* and *who*. While a comma preceding the former has most likely a coordinating function, a comma following the latter most likely does not. We include the window of *n* tokens and POS tags directly adjacent to the punctuation terminal as features. We also take the normalized position of the punctuation sign, i.e., its position index normalized by sentence length, as indicator for its function, as well as the respective normalized positions of the next coordinating/non-coordinating punctuation signs on its left and right. Additionally, we use the relative distance of the next coordinating conjunction (CC) as information since most coordinations have an overt conjunction.

To describe the syntactic context of a punctuation sign, we first consider the label of its parent node since this should be the node to which the conjuncts to the left and to the right of the punctuation are attached. We add the siblings of the parent, the grandparent, and information whether the parent dominates a coordinating conjunction. We also include *leaf-ancestor (LA) paths* [19] of the *n* words and POS tags to the left and right of the punctuation. An LA path is the concatenation of labels encountered on the path from a leaf node to the root node (including both). We modify LA paths so that they stop at the parent of the punctuation sign since the global context should not provide much relevant information.

4.2 Parsing

The syntactic features can be obtained directly from the gold treebank trees. However, in order to provide a more realistic setting, we also investigate the effect of obtaining them from the trees output by a parser. In order to build a parsed version of the training set of the classifier, we use the Berkeley parser [18]. In order to avoid parsing on data seen in training, we use jackknifing on a 5-fold setting. We use the default settings for the Berkeley parser. The results for the concatenated training set and the test set are 84.92/85.03/84.98 in terms of precision, recall, and F-score.

4.3 Evaluation

Since we treat our problems as a binary classification problem, the obvious evaluation metric would be accuracy. However, the sets are heavily skewed towards negative examples, and many of those are clearly non-coordinating, such as commas before `who`. For this reason, accuracy would place an overly high weight on those negative cases. Thus, we evaluate the classifier performance with regard to both classes, i.e., coordination or non-coordination. We compute precision, recall (the true positive rate), the F-score, the false positive rate. We report significance based on McNemar's test, on the 0.01 and the 0.1 level.

extremely low, even after sampling to reduce the skewing, we concentrate on TiMBL in this paper.

	positive				negative			
	prec.	recall	fp-rate	F-score	prec.	recall	fp-rate	F-score
baseline (pos)	87.31	80.34	2.12	83.68	96.49	97.88	19.66	97.18
pos + focus	88.99	80.63	1.81	84.60	96.55	98.19	19.37	97.36
pos + focus additionally:								
+1r	89.93	76.35	1.55	82.59*	95.83	98.45	23.65	97.12
+2r	90.88	76.64	1.39	83.15	95.88	98.61	23.36	97.23
+3r	90.16	78.35	1.55	83.84	96.17	98.45	21.65	97.30
+1l	88.16	80.63	1.96	84.23	96.54	98.04	19.37	97.28
+2l	88.25	79.20	1.91	83.48	96.30	98.09	20.80	97.19
+3l	89.03	78.63	1.75	83.51	96.21	98.25	21.37	97.22
+1l+1r	90.00	79.49	1.60	84.42	96.36	98.40	20.51	97.37
+2l+2r	90.82	78.92	1.45	84.45	96.27	98.55	21.08	97.40
+3l+3r	91.18	79.49	1.39	84.93	96.37	98.61	20.51	97.47
pos + focus + 3l + 3r additionally:								
+cc	90.25	81.77	1.60	85.80	96.75	98.40	18.23	97.57
+dist	91.83	80.06	1.29	85.54	96.47	98.71	19.94	97.58
+neigh	**92.77**	**84.05**	1.19	**88.19****	**97.16**	**98.81**	15.95	**97.98**
+dist+neigh	92.43	83.48	1.24	87.72*	97.06	98.76	16.52	97.90
+cc+dist+neigh	90.40	83.19	1.60	86.65	97.00	98.40	16.81	97.69

Table 2: Results for local information, with significance tested against baseline for pos + focus, against pos + focus for all other experiments; *=0.1; **=0.01.

5 Results

In our test set, we have 2 288 instances of punctuation. Out of those, 351 are annotated as coordinating. This means that by classifying all punctuation as non-coordinating, we reach an accuracy of $\frac{2\,288-351}{2\,288} = 84.66$. However, in our experiments, we are particularly interested in achieving a high F-score with respect to coordinating punctuation, i.e., the positive class.

To determine parameter settings, we conducted a non-exhaustive search and found the IB1 algorithm with overlap metric for the computation of instance distances, feature weighting via GainRatio, using the $k = 5$ nearest neighbors, and an inverse linear weighting of neighbors as function of their distances to be the best parameter combination (see [3] for an explanation of the parameters). All experiments reported below are based on these settings.

5.1 Using Local Context

Our baseline consists of three POS tags to the left and to the right of the punctuation sign. See the first row of table 2 for the results. Next, we add the punctuation sign itself (+ focus). This results in an increase of almost one percent point in terms of positive F-score, as shown in row two.

A separate evaluation of commas and semicolons for the experiment with baseline and focus word shows a positive F-score of 82.56 for commas and 96.23 for

semicolons, compared to 84.60 for the complete evaluation.[3] Semicolons are classified much more reliably, most likely due to the fact that most of them are used on the clausal level, where only coordination cases with overt conjuncts are annotated. Thus, the most difficult cases, particularly those which involve a semantic component (such as apposition separator vs. NP coordination), involve commas. Also note that there are many more commas (2 143) than semicolons (76).

We now consecutively add lexical information in order to test our intuition of its discriminative power. The next part of table 2 shows the results. The highest F-score in this part is obtained by using 3 words to the left and to the right (+ 3l + 3r): a positive F-score of 84.93, which results from a high precision of 91.18. However, this increase is not significant over the pos + focus setting. Generally, adding context words increases precision, but has a larger detrimental effect on recall.

In the next section of the table, we add the position of the punctuation sign within the sentence (+ dist), the positions of neighboring (coordinating and non-coordinating) punctuation signs (+ neigh) to the left and right (i.e., 4 features), and the position of the next conjunction to the right (+ cc). The information about the position of the next `CC`-tagged word on the right (+ cc) and the position of the punctuation sign within the sentence (+ dist) surprisingly do not lead to a significant improvement of the F-score, as intuition would suggest. However, the four features of + neigh are particularly effective, leading to a significantly higher F-score. This setting reaches the highest overall results: positive precision: 92.77, recall: 84.05, and F-score: 88.19, negative precision: 97.16, recall: 98.81, and F-score: 97.98.

5.2 Using Gold Syntax

In this section, we investigate the benefit of adding (gold) syntactic features. We first add the parent node (+ p) of the punctuation sign. Then we add the grandparent (+ gp) and the two directly adjacent sibling non-terminals (+ sib) of the parent. In a next step, we add leaf-ancestor paths (+ la) as well as information whether the parent dominates a coordinating conjunction (+ ccn). As further features, we use the punctuation sign itself (+ focus), the position of its neighbors (+ neigh), and the three words around it (+ 3l3r). All syntactic features are extracted from the Penn Treebank.

Table 3 presents the results when using gold syntax.[4] The first row repeats the baseline of three POS tags left and right of the punctuation sign (here with gold POS tags). When we add the parent node (+ p), we gain approximately 2.7 percent points in positive precision and about 4.5 percent points in recall. This results in an F-score of 88.09, significantly higher than the baseline, and only 0.1 lower than the F-score we gained by adding all the lexical context, thus showing how important the syntactic context of the punctuation sign is. Adding the grandparent

[3] Here, we ignore { – - . . . } because there are no positive instances of them in the test set.

[4] Note that the baseline results are better than for the local context since in the latter, we have used the POS tags created by the Berkeley parser.

	positive				negative			
	precision	recall	fp-rate	F-score	precision	recall	fp-rate	F-score
baseline (pos)	89.46	79.77	1.70	84.34	96.41	98.30	20.23	97.34
+p	92.21	84.33	1.29	88.09**	97.20	98.71	15.67	97.95
+p+gp	92.03	85.47	1.34	88.63	97.40	98.66	14.53	98.02
+p+gp+sib	91.87	86.89	1.39	89.31	97.65	98.61	13.11	98.13
+ p + gp additionally:								
+la	90.73	80.91	1.50	85.54*	96.61	98.50	19.09	97.55
+sib+la	90.69	86.04	1.60	88.30	97.49	98.40	13.96	97.95
+ccn	93.98	88.89	1.03	91.36**	98.01	98.97	11.11	**98.48**
+sib+ccn	93.43	89.17	1.14	91.25**	98.05	98.86	10.83	98.46
+ p + gp + sib + ccn additionally:								
+focus	93.16	**89.17**	1.19	91.12	98.05	98.81	10.83	98.43
+focus+3l3r	92.49	87.75	1.29	90.06*	97.80	98.71	12.25	98.25
+focus+neigh	93.71	**89.17**	1.08	91.39	**98.06**	98.92	10.83	98.48
all features	**94.26**	88.89	0.98	**91.50**	98.00	**99.02**	11.11	**98.51**

Table 3: Results for gold syntax. Significance: against baseline for + p, against + p for first/second experiments, against + ccn for third/fourth set; *=0.1; **=0.01.

adds an additional 1.1 percent points to recall, which reaches 85.47, but results in a minimal decrease of precision, causing no significant change in F-score. We benefit from including the siblings of the parent node of the punctuation sign: the positive F-score increases slightly, though not significantly.

Adding the leaf-ancestor path to the parent + grandparent combination causes a significant decrease of the F-score, as shown in the second part of the table. Adding it to the combination including siblings does increase the F-score to 88.30. However, this score is lower than the one using sibling information alone. As expected, + ccn is more important, these results significantly improve over the + p setting. Combining it with sibling information gives no significant difference.

In the third part of the table, we add information from the previous section that proved helpful: the focus punctuation sign, its neighbor punctuations, as well as the lexical context. When we add the focus to the combination baseline + parent + grand-parent + siblings + `CC`-daughter of parent, we reach the highest positive recall overall, 89.17, but the F-score does not improve significantly over + ccn. Adding all the information results in the same recall of 88.89 and in the highest precision (94.26), as well as in the highest positive F-score overall (91.50). These experiments show that crafting a successful feature combination is a difficult task; and they indicate that lexical features seem to partially provide a syntactic context when there is no syntactic information. But the syntactic information is more reliable when available.

	positive				negative			
	precision	recall	fp-rate	F-score	precision	recall	fp-rate	F-score
baseline (pos)	87.31	80.34	2.12	83.68	96.49	97.88	19.66	97.18
+p	90.18	83.76	1.65	86.85**	97.09	98.35	16.24	97.72
+p+gp	91.33	84.05	1.45	87.54	97.15	98.55	15.95	97.85
+p+gp+sib	89.91	83.76	1.70	86.73	97.09	98.30	16.24	97.69
+ p + gp additionally:								
+la	89.84	80.63	1.65	84.98*	96.55	98.35	19.37	97.44
+sib+la	88.29	83.76	2.01	85.97	97.08	97.99	16.24	97.53
+ccn	91.02	83.76	1.50	87.24	97.10	98.50	16.24	97.80
+sib+ccn	90.21	84.05	1.65	87.02	97.14	98.35	15.95	97.74
+ p + gp + sib + ccn additionally:								
+focus	90.91	85.47	1.55	88.11	97.39	98.45	14.53	97.92
+focus+3l3r	92.09	82.91	1.29	87.26	96.96	98.71	17.09	97.83
+focus+neigh	92.64	**86.04**	1.24	**89.22***	**97.50**	98.76	13.96	**98.13**
all features	**93.40**	84.62	1.09	88.79*	97.26	**98.92**	15.39	98.08

Table 4: Results using the Berkeley parser with significance tested against baseline for + p, against + p for all other experiments; *=0.1; **=0.01.

5.3 Using Parsing

Here, we investigate whether we can replicate the results above with syntactic information from the Berkeley parser. Table 4 presents the results based on parser information. For direct comparison, we report the same feature sets from table 3.

We see that some general trends are repeated: Adding the parent, grandparent, and focus information is helpful, adding the leaf-ancestor paths is not. Using gold syntax and baseline features results in an F-score of 84.34; the same experiment using parsing data results in an F-score of 83.68. Major decreases in positive F-score occur only with features that heavily rely on syntactic information, such as + ccn. However, the highest positive F-score achieved with the gold syntax (91.50) is just 2.2 points higher than the one based on parser data.

The lower quality of parse trees is particularly reflected in two features: Features concerning the siblings (+ sib), and, much more so, the parent of a `CC` daugther (+ ccn), are less reliable than their gold syntax counterparts. Instead, we now benefit more from the lexical material around the focus punctuation. The highest positive recall (86.04) and F-score (89.22) is reached by using parent + grandparent + sibling + focus + neighbors + `CC`-daughter information.

We now look at the results with best results from table 4 (+ p + gp + sib + ccn + foc + neigh), and we separate them based on the mother node. Table 5 lists the results for the most frequent categories (all other categories had considerably fewer instances in the test set). Coordinations under S nodes can be detected very reliably, probably because only overt conjunction cases are annotated, and these are the easier cases. NPs and VPs are more difficult because they include a larger range of phenomena: covert conjunction cases, appositions, and superfluous conjunction

	#	precision	recall	F-score
S	735	97.12	94.39	95.73
VP	355	87.50	83.05	85.22
NP	912	90.84	83.80	87.18

Table 5: The results of the +p + gp + sib + ccn + foc + neigh experiment based on the type of mother node.

commas. However, further insights require a more detailed error analysis.

6 Conclusion and Future Work

We have presented work on automatically distinguishing punctuation signs functioning as separators for conjuncts in coordination from such that do not have a coordinating function. We used a version of the Penn Treebank in which punctuation signs are annotated whether they have a coordinating function [14]. We have formulated the task of identifying the status of previously unseen punctuation as a binary classification problem. Using memory-based learning, we have achieved an F-score of 91.50 (98.51) on positive (negative) instances of punctuation using features drawn from the local context of the punctuation sign to be classified and from its surrounding syntactic context given by the treebank annotation. Even when using syntactic trees provided by a parser, i.e., without any gold information, we still achieve a F-score for positive (negative) instances of 89.22 (98.13).

Our experiments are a first step towards a reliable, cross-category identification of coordination with scope detection using supervised learning. Our goal is to include all types of coordination, even if no overt conjunction is present. We are also planning to include the information learned here into a parsing approach to improve parsing across all types of coordination.

References

[1] Shane Bergsma, David Yarowsky, and Kenneth Church. Using large monolingual and bilingual corpora to improve coordination disambiguation. In *ACL*, 2011.

[2] Francis Chantree, Adam Kilgarriff, Anne de Roeck, and Alistair Willis. Disambiguating coordinations using word distribution information. In *RANLP*, 2005.

[3] Walter Daelemans, Jakub Zavrel, Ko Van der Sloot, and Antal Van den Bosch. TiMBL: Tilburg Memory Based Learner, version 6.3, Reference Guide. ILK Research Group Technical Report Series 10-01, University of Tilburg, 2010.

[4] Robert Dale and Paweł Mazur. Handling conjunctions in named entities. In *Computational Linguistics and Intelligent Text Processing*, volume 4394 of *LNCS*, pages 131–142. Springer, 2007.

[5] Yuqing Guo, Haifeng Wang, and Josef van Genabith. Recovering non-local dependencies for Chinese. In *EMNLP-CoNLL*, 2007.

[6] Kazuo Hara, Masashi Shimbo, Hideharu Okuma, and Yuji Matsumoto. Coordinate structure analysis with global structural constraints and alignment-based local features. In *ACL-AFNLP*, 2009.

[7] Katharina Hartmann. *Right Node Raising and Gapping*. John Benjamins, 2000.

[8] Enrique Henestroza Anguiano and Marie Candito. Parse correction with specialized models for difficult attachment types. In *EMNLP*, 2011.

[9] Deirdre Hogan. Coordinate noun phrase disambiguation in a generative parsing model. In *ACL*, 2007.

[10] Thorsten Joachims. SVM^{light} support vector machine, 2002. `http://svmlight.joachims.org`.

[11] Daisuke Kawahara and Sadao Kurohashi. Generative modeling of coordination by factoring parallelism and selectional preferences. In *IJCNLP*, 2011.

[12] Adam Kilgarriff, Pavel Rychlý, Pavel Smrž, and David Tugwell. The Sketch Engine. In *EURALEX*, 2004.

[13] Sandra Kübler, Erhard W. Hinrichs, Wolfgang Maier, and Eva Klett. Parsing coordinations. In *EACL*, 2009.

[14] Wolfgang Maier, Sandra Kübler, Erhard Hinrichs, and Julia Krivanek. Annotating coordination in the Penn Treebank. In *ACL-LAW VI*, 2012.

[15] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Comp. Ling.*, 19(2):313–330, 1993.

[16] Andrew Kachites McCallum. MALLET: A machine learning for language toolkit, 2002. `http://mallet.cs.umass.edu`.

[17] Philip Ogren. Improving syntactic coordination resolution using language modeling. In *NAACL HLT SRW*, 2010.

[18] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *COLING-ACL*, 2006.

[19] Geoffrey Sampson and Anna Babarczy. A test of the leaf-ancestor metric for parse accuracy. *Journal of Natural Language Engineering*, 9:365–380, 2003.

TüNDRA: A Web Application for Treebank Search and Visualization

Scott Martens

Seminar für Sprachwissenschaft
Eberhard Karls Universität Tübingen
E-mail: scott.martens@uni-tuebingen.de

Abstract

This paper describes the design and use of TüNDRA: the Tübingen aNnotated Data Retrieval and Analysis tool. TüNDRA is a web application for searching in treebanks of all varieties, visualizing their contents, and extracting basic statistics from them. It uses a web-based model for corpus software in which users do not have to install, upgrade or support complex tools, or deal with multiple data formats. Providing treebank tools in this manner maximizes flexibility and functionality while dramatically increasing usability by making data immediately available to broad communities of researchers.

1 Introduction

TüNDRA is a novel web application for browsing, searching and visualizing the contents of treebanks, accessible to users via the CLARIN single-sign-on digital resource infrastructure (Ketzan & Schuster [5]). It has a number of important benefits for treebank research:

- TüNDRA places very few restraints on the kinds of annotations it supports. Treebank nodes, whether they correspond to tokens, words, constituents or other abstractions, support any kind of feature with a string value. Dependencies and relations between tree elements can have labels.
- TüNDRA supports both constituency and dependency treebanks, as well as mixed and hybrid treebank types. Any pair of elements in a tree can be connected by an edge. There is no requirement for syntax trees to be projective and there is limited support for directed graphs, although TüNDRA is oriented primarily towards tree-structured annotation.
- TüNDRA supports all languages and writing schemes for which a freely usable WOFF font[1] is available, including right-to-left oriented scripts like Arabic, as well as IPA notations.

[1] Web Open Font Format: Essentially, a wrapper around a TrueType or OpenType font.

- TüNDRA is offered as a web application, which provides important benefits for accessibility and a very different use and support model from conventional software.

2 Related work

A number of tools are available for searching and displaying treebanks. Of particular interest for this project are the TIGERSearch (Lezius [6]) and TrEd (Pajas and Štěpánek [10]) suites. Furthermore, ANNIS[2], GrETEL[3], and INESS[4], currently offer treebank search and visualization as a web application, similar to TüNDRA.

TIGERSearch is free, Java-based software that runs on desktop computers. It only supports constituency treebanks and offers little support for non-European languages. It is, nonetheless, a popular suite for treebank research, and TüNDRA has been planned, from the outset, to meet the needs of TIGERSearch's current users.

TrEd supports many varieties of treebanks, like TüNDRA, but has a complex data format into which all data must be converted. Conversion scripts are available for some standard treebank data formats. Users can configure treebank visualizations using a stylesheet language unique to TrEd.

TIGERSearch and TrEd both require users to download, install and configure desktop software. This can be a heavy burden for users and system administrators, but is intrinsic to the desktop software model (Henrich et al. [3]).

ANNIS (Zeldes et al. [14]) is web-based, supports diverse kinds of treebanks and uses a query language similar to TIGERSearch, but is offered as software to download and install locally rather than as a remotely hosted web application. It also has the ability to import treebanks in a variety formats, but the procedure can be quite complex. Like TüNDRA, ANNIS uses a modified form of the TIGERSearch query language to search treebanks.

GrETEL (Augustinus et al. [1]) provides a web-based query interface and treebank visualization service for a selection of Dutch-language treebanks. It presently only supports constituency treebanks, although it allows treebank edges to have labels, and has no direct support for dependency grammar or non-tree edges.

INESS (Rosén et al. [11]) is the closest to TüNDRA in intent and functionality. It is a web-based application that supports a similar array of treebanks, and uses an adaptation and extension of the TIGERSearch query language. INESS has substantially greater visualization abilities for LFG and HPSG style annotation than TüNDRA as well as support for parallel treebanks, which is currently not a feature of TüNDRA. Furthermore, INESS offers users access to the XLE parser[5] for a number of languages, so that they can create treebanks from inputted text.

[2] http://www.sfb632.uni-potsdam.de/annis/

[3] http://nederbooms.ccl.kuleuven.be

[4] http://iness.uib.no

[5] http://www2.parc.com/isl/groups/nltt/xle/

TüNDRA is a part of the CLARIN-D project[6], which is developing an integrated scalable infrastructure for the social sciences and humanities with funding from the German Federal Ministry for Education and Research (BMBF). It is integrated with CLARIN-D's WebLicht language tool chaining environment (Hinrichs et al. [4]), enabling users to create treebanks from texts using a variety of tools, as well as providing a broad array of services to researchers for developing and distributing treebanks and other linguistic resources.

3 Treebank access as a web application

TüNDRA is offered as a web application – an alternative approach to software development and distribution that has a number of advantages over traditional software[7]:

- A web application is accessible from any compatible browser, on any computer with an adequate Internet connection. There is nothing to download, install or configure, and no particular technical expertise required.
- New features, changes, and bug fixes are available to users without reinstalling any software. Tools made available as web applications can be in a constant state of improvement without interfering with users' activities.
- Treebank size and access speeds are not bounded by a desktop computer's limited memory and storage.
- Users do not have to import treebanks from the many different treebank data formats, and problems of software compatibility never arise for treebanks available via TüNDRA.

The principal drawback of providing treebank access as a web application relates to the last point. With locally run software, users can install any compatible treebank they possess. With TüNDRA, the treebank must be hosted on a remote server and treebank data must reside there. This creates a number of new issues:

- Treebanks may have licensing restrictions incompatible with web distribution and access.
- Introducing new treebanks into TüNDRA involves work for the hosting entity, which must take charge of data conversion and any compatibility issues that arise.

To address the first point, the CLARIN infrastructure offers a number of mechanisms to restrict data access to only qualified users, instead of everyone with access to the infrastructure. TüNDRA's currently supported treebanks are accessible

[6] `http://de.clarin.eu/`

[7] See [3] for further elaboration on the motivation behind web applications.

for all academic users, but the capability exists to further restrict access to specific treebanks. Nonetheless, for many popular treebanks, we can only offer them through specially negotiated licenses, and only if the owners are willing.

For the second point, there are two solutions we have implemented or are implementing as part of the TüNDRA web application. First, for owners of treebanks who are willing to make their work available to the academic community via the Tübingen CLARIN data repository (Dima et al. [2]), we are provisionally prepared to take charge in-house of converting them to TüNDRA's internal data format and making whatever changes are necessary to TüNDRA to ensure their correct display and accessibility. We believe this tool can be very valuable for treebank researchers and are committed to offering a diverse selection of treebanks to the research community. Furthermore, offering TüNDRA as a web application means that if changes have to be made to the software to adequately offer a treebank to users, we can introduce those changes without having to distribute new editions.

Second, TüNDRA is integrated into WebLicht's expanding functionality for importing and exporting various data formats, and instead of introducing data conversion functionality directly into TüNDRA, we will provide data conversion and user storage through WebLicht. This gives users the ability to upload treebanks and use them on demand without distributing them.

4 TüNDRA web interface design

TüNDRA uses a single screen so that users never see an entire page disappear and be replaced by something else. Although provided as a web page in a browser, it functions much more like a traditional GUI application, and endeavours to uphold the venerable *Principle of Least Surprise* in software interface design. Sudden, unexpected screen changes never take place, and in so far as is feasible, buttons and menus have standard names and icons, and do what similar controls do in other software. Comprehensive documentation of the interface is available at all times, and on-screen messages and help facilities provide usage information wherever it can be introduced without interfering with user activity.

TüNDRA's design has been influenced by elements of *Activity-Centred Design* (Nardi [9]), which shifts the emphasis in design away from studies of user preferences and instead focuses on users' activities and contexts. Its principal goal is to make it as easy as possible for users to perform their tasks so that they can quickly adapt to the tool, rather than making the tool adapt to users' preconceptions about what it should do. This approach is especially well-suited to treebank applications because TüNDRA's users will likely already have some familiarity with treebanks and syntactic theories, and many will have some experience with other corpus tools. These tools are very diverse and a single application cannot incorporate all those preconceptions without producing an unwieldy and awkward interface. TüNDRA's multiple tree visualizations and search language draw heavily on existing sources, but the interface is a new design that attempts to make the tasks that we expect users to want to perform as immediately accessible as possible.

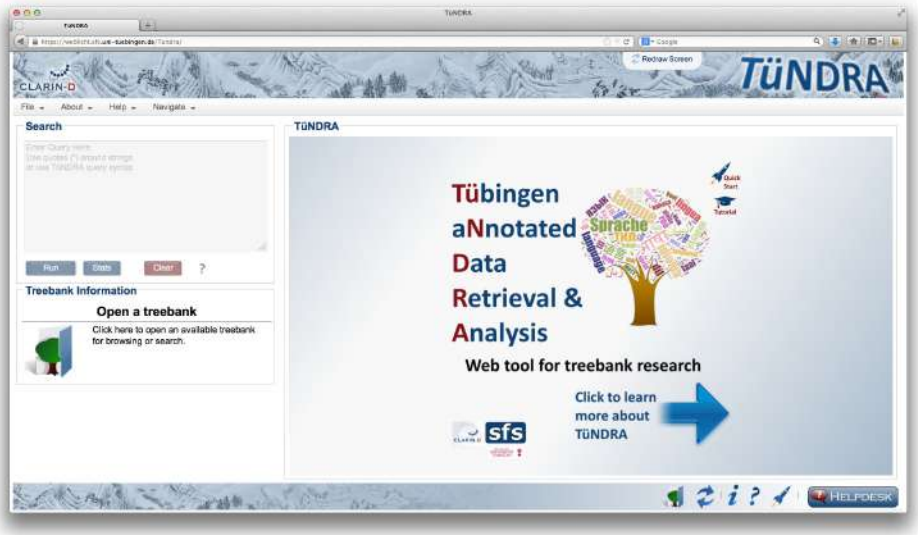


Figure 1: TüNDRA start screen displayed in the Firefox browser

As a specific example of how this approach has influenced TüNDRA's design, there is special functionality that appears when TüNDRA is used in a browser with a small area, providing a more compact display that is especially suited for digital projectors. Enquiries into the purposes for which users employ treebank visualization software revealed that classrooms and conferences are very important. Usability at small resolutions is, therefore, a high priority function for TüNDRA.

5 Using TüNDRA

Users from academic institutions participating in the DFN-AAI credential service[8] automatically have access to TüNDRA through the CLARIN-D infrastructure using their existing institutional login credentials. Other academic users can request an account directly from CLARIN-D[9]. Users can access TüNDRA at `https://weblicht.sfs.uni-tuebingen.de/Tundra/`.

Figure 1 is the opening screen users see on logging into TüNDRA. This screen is designed to encourage users to inform themselves at least a little bit before diving into TüNDRA, while allowing experienced users to move directly to the content without having to dismiss popups or other irritants at start time. Users can open an installed treebank immediately for inspection and search by clicking the icon marked "*Open a treebank*" and then selecting from a list of installed treebanks. Users can then browse freely, access a sentence by index number, or enter in the upper left corner textbox a treebank pattern for TüNDRA to match (using the search syntax described in Section 6) and view the results.

[8] `https://www.aai.dfn.de/`

[9] `https://user.clarin.eu/user/register`

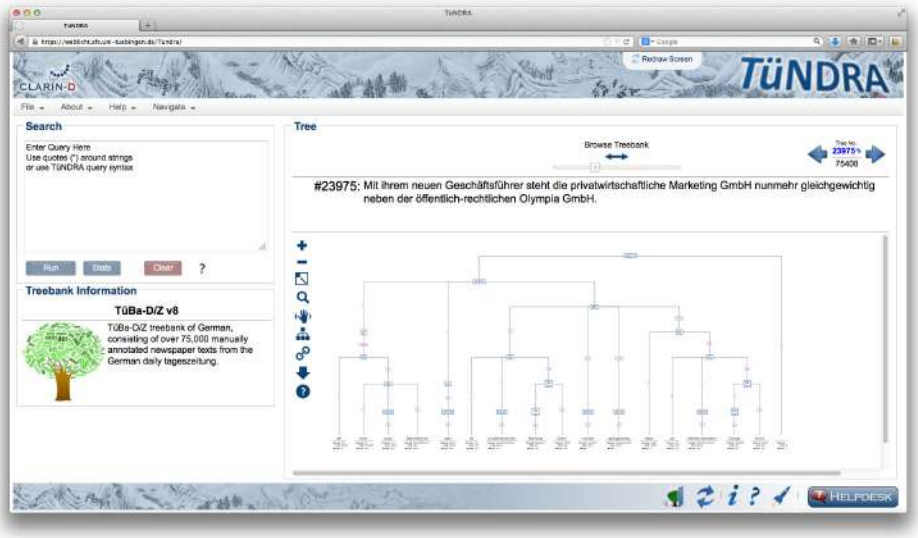


Figure 2: TüNDRA displaying a tree from the TüBa-D/Z treebank of German

Trees and query results are displayed in the large panel on the right of the screen, as in Figure 2. The top part of the tree visualization panel shows information about the current sentence, its place in the treebank or query results, and has navigation controls for viewing other sentences in the treebank.

6 Searching

TüNDRA was designed from the outset to reimplement the TIGERSearch query language (Martens [7]). TüNDRA implements all TIGERSearch functionality except for user-defined shortcuts, and has extended the language in some areas. Users familiar with TIGERSearch will find its behaviour almost identical. The most important difference is in the treatment of negated relations, discussed in Section 6.2.

This section offers a very cursory look at TüNDRA query syntax, and much richer information is available on logging into TüNDRA. All example queries in this section are based on the annotation scheme of the TüBa-D/Z treebank of German (Telljohann et al. [13]), which is installed and searchable through TüNDRA.

6.1 Features

To match a constituent or token with a particular feature value, users enter a query like the one below:

```
[pos="NN"]
```

This query matches all elements with a feature named `pos` whose value is `NN` – in the example corpus, any token whose part-of-speech corresponds to an ordinary noun. Specifications for multiple features simply join them with the boolean `&` (AND) and `|` (OR) operators.

Feature names and values can be any Unicode string, and any node in a tree can have as many features as required. Feature names are arbitrary, except for a few features with special display properties or querying functions.

TüNDRA supports Unicode regular expressions searches on all labels. For example, to find all tokens containing the character "ß", use `[word=/.*ß.*/]`.

6.2 Relations

To query relations between two nodes, two node specifications are connected using a relational operator. For example, to query for a node whose `cat` attribute has the value `NX` (the standard label for noun phrases in the TüBa-D/Z treebank) that is also the immediate parent of another node with a `pos` attribute that has the value `NN`, use:

```
[cat="NX"] > [pos="NN"]
```

When a parent-child relation has a label, that can also be queried.

Similarly, to search for a node matching `[cat="NX"]` that is an ancestor of a node matching `[pos="NN"]` at any distance:

```
[cat="NX"] >* [pos="NN"]
```

And to query for nodes matching `[cat="NX"]` that do not have any immediate descendants that match `[pos="NN"]`:

```
[cat="NX"] !> [pos="NN"]
```

Users familiar with TIGERSearch should note that negated relations are handled differently in TüNDRA. In TIGERSearch, whenever a query specifies a node, even in negation, it must match some node in a matching sentence, reducing the usefulness of operator negation in TIGERSearch.

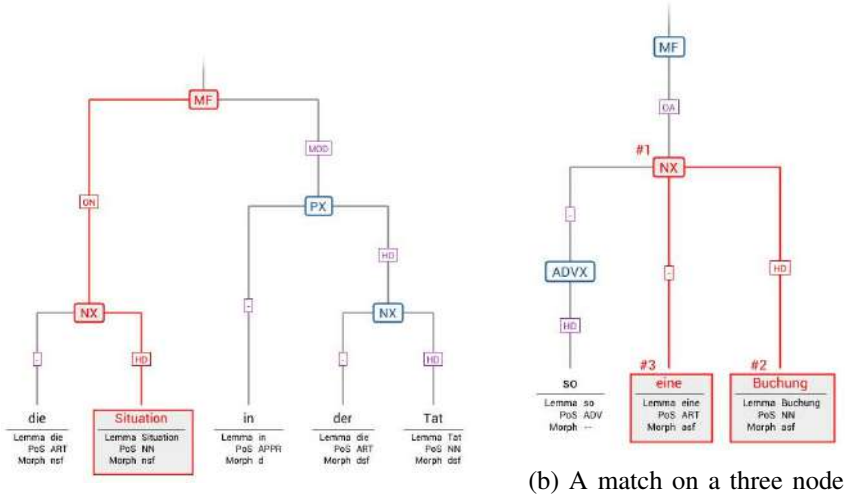
In TüNDRA, negated relations more closely match user expectations. The query above will match nodes of category `NX` even in sentences that do not contain any node matching `[pos="NN"]`. However, full universal negation - negating entire structures instead of individual relations - has not yet been implemented directly in TüNDRA syntax.

Relations of several kinds are available for querying using a similar syntax. For example, to query for the words "the" and "bank" next to each other:

```
[word="the"] . [word="bank"]
```

Searching for secondary edges (additional relations that are not part of the main syntactic tree, like co-references) and their labels is also supported, using the same query syntax as TIGERSearch (the `>~` operator), without restrictions on the nodes that can bear secondary edges. This provides limited support for non-tree structures. Full support for arbitrary directed graphs is possible using the TüNDRA query engine, but this has not yet been implemented in the interface.

TüNDRA supports queries for ancestry and descent, surface sequence (i.e. word and constituent order), sibling relations, and siblings in surface sequential order.



(a) A match on a three node parent-child structure

(b) A match on a three node structure with multiple constraints

Figure 3: Querying multiple nodes and complex structures

6.3 Variables and multiple relations

To search for more complex structures, TüNDRA query syntax supports chaining together relations on nodes, and supports variable labels so that one node can participate in multiple relations. For example, the two queries below produce identical search results and match the structure in Figure 3a:

```
[cat="MF"] > [cat="NX"] > [pos="NN"]
[cat="MF"] > #1:[cat="NX"] & #1 > [pos="NN"]
```

Using variables makes it possible to query on structures that involve, for example, multiple descendants of a single node, or different kinds of relations on a single node, or with multiple constraints, as in the query below, matching Figure 3b :

```
#1:[cat="NX"] > #2:[pos="NN"]  & #1 > #3:[pos="ART"] & #3 . #2
```

Variables can also be used for nodes without specifying any features. For example, the query `#1 > [word="Band"]` identifies all parents of a node matching `[word="Band"]`, without any restriction on their features.

6.4 Statistics

Variables are also used to collect statistics on matches. For example, to get counts for all the features (including tokens, lemmas, and morphological information) of all grammatical articles in the TüBa-D/Z treebank that immediately precede the word "Band", use the following query:

```
#1:[pos="ART"] . [word="Band"]
```

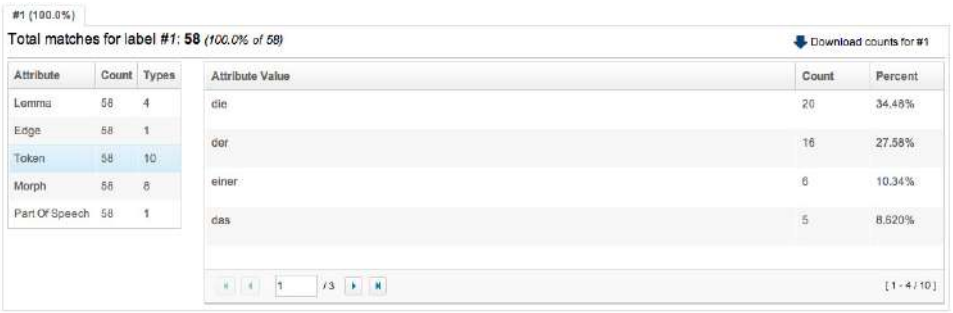
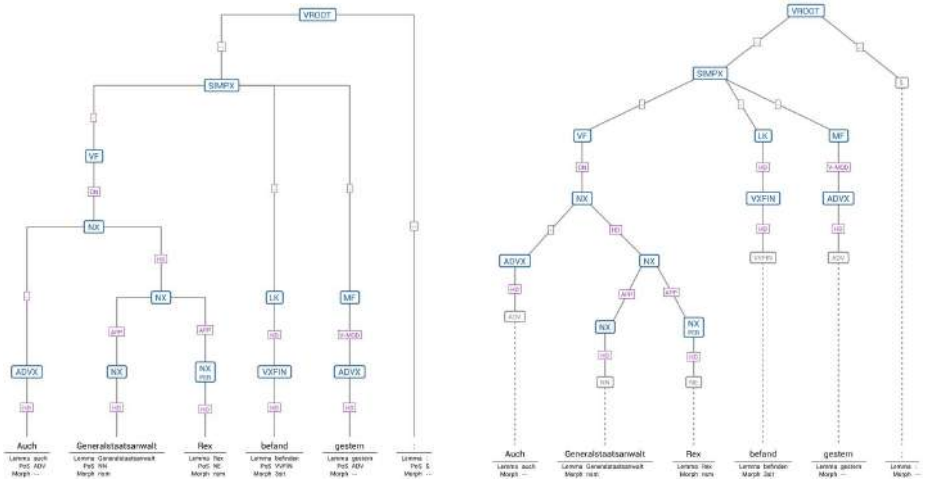


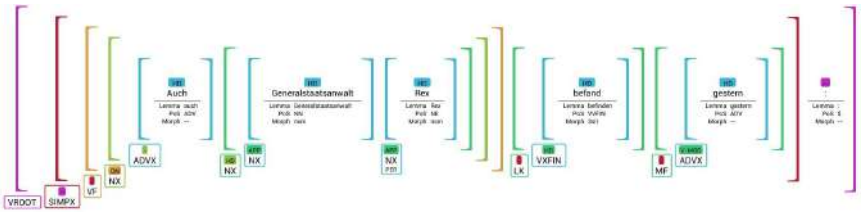
Figure 4: Statistics results

Note that the node specification is accompanied by a variable name. This would not be necessary for simply searching for matching structures, but is required to collect statistics. The results are displayed on screen, as in Figure 4. Users can browse, reorder, and download the results for further processing.



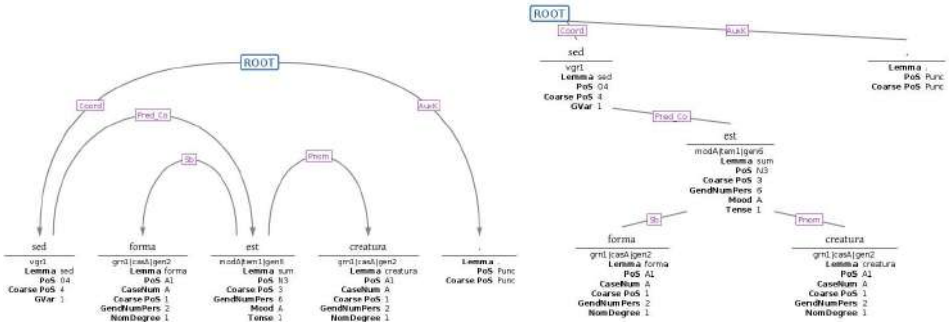
(a) Tiger style

(b) Generative grammar style



(c) Bracketed style

Figure 5: Three visualizations of a constituency tree from the TüBa-D/Z treebank



(a) Word Grammar style (b) TrEd style

Figure 6: Two visualizations of a dependency trees from the *Index Thomisticus Treebank* (McGillivray et al. [8]), accessible via TüNDRA.

7 Visualization

In order to provide the greatest immediate utility to existing users of treebank software, TüNDRA provides three different visualization schemes for constituency trees and two for dependency trees. Figure 5 shows all three constituency tree visualizations of the same sentence from the TüBa-D/Z:

1. TIGERSearch style tree display. (Figure 5a)
2. A generative grammar style tree display. (Figure 5b)
3. A linear display with constituents designated by brackets, using different sizes and colours of brackets to improve readability. (Figure 5c)

Similarly, TüNDRA has two tree drawing styles for dependency trees:

1. A linear display with arcs between words, based on the styles frequently used for Word Grammar (Sugayama & Hudson [12]) and some other dependency formulations. (Figure 6a)
2. A drawing style based on the TrEd software (Pajas and Štěpánek [10]) frequently used with the Prague Treebank and similar corpora. (Figure 6b)

All trees are drawn to match the surface order of tokens. Non-projective relations are displayed as crossing branches. TüNDRA does not impose projectivity on the trees it displays.

On-screen controls allow users to switch freely between visualizations, to magnify and shrink all or parts of the display, navigate in the visualization, and download the rendered visualizations in several graphics formats.

8 Technical implementation

TüNDRA uses a number of off-the-shelf web application frameworks and technologies, but is particularly reliant on browser support for SVG[10]. Consequently, users with older or non-standard browsers may not be able to use TüNDRA. Using SVG makes it easier to let users download rendered trees for their papers and presentations - a major feature for academic researchers.

The TüNDRA query system is built around the open source BaseX[11] library for XQuery[12]. TüNDRA queries are translated into XQuery syntax, then applied to treebanks that are encoded in an application-specific XML tagset. Using BaseX as an underlying framework has made it much easier to develop and extend TüNDRA, since the lower level requirements of a query resolution system are already handled by a robust, fast, externally maintained library. XQuery is a sufficiently powerful language that any extension to TüNDRA can be implemented by simply identifying appropriate XQuery syntax.

9 Conclusion

TüNDRA is a new approach to treebank search and visualization software, one intended to support a broad array of treebanks including both constituency and dependency formalisms. It operates as independently as possible of the linguistic assumptions that underlie individual treebanks. TüNDRA also eliminates many difficult problems with software installation, data formats and resource limitations.

TüNDRA is an on-going project, and by opening it up to as large an audience as possible, we hope to receive feedback from researchers in order to make it progressively more useful. The web application model makes it more than just a piece of software – it is also a constantly improving publishing platform for treebanks. We are committed to maintaining and extending TüNDRA as a medium for making treebanks more accessible to the research community.

References

[1] Augustinus, L., V. Vandeghinste, and F. Van Eynde (2012). Example-Based Treebank Querying. In: *Proc. of the 8th International Conference on Language Resources and Evaluation* (LREC 2012). Istanbul.

[2] Dima, E., V. Henrich, E. Hinrichs, M. Hinrichs, C. Hoppermann, Thorsten Trippel, Thomas Zastrow and Claus Zinn (2012) A Repository for the Sustainable Management of Research Data. In: *Proc. of the 8th International Conference on Language Resources and Evaluation* (LREC 2012), Istanbul.

[10] *Scalable Vector Graphics*, a W3C standard for vector drawing in web browsers.

[11] `http://basex.org/`

[12] A W3C standard for searching and processing XML documents.

[3] Henrich, V., E. Hinrichs, M. Hinrichs and T. Zastrow (2010) Service-Oriented Architectures: From Desktop Tools to Web Services and Web Applications. In: *Multilinguality and Interoperability in Language Processing with Emphasis on Romanian*, Romanian Academy, Bucharest, pp. 69-92.

[4] Hinrichs, E., M. Hinrichs, and T. Zastrow (2010) WebLicht: web-based LRT services for German. In: *Proc. of the ACL 2010 System Demonstrations* (ACLDemos '10), Stroudsburg, PA, USA, pp. 25-29.

[5] Ketzan, E. and I. Schuster (2012) CLARIN-D User Guide, Ch. 4: Access to resources and tools – technical and legal issues. In: *CLARIN-D User Guide v. 1.0.1*, pp. 41-45. (URL: http://media.dwds.de/clarin/userguide/text/)

[6] Lezius, W. (2002) TIGERSearch – Ein Suchwerkzeug für Baumbanken. In: *Proc. der 6. Konferenz zur Verarbeitung natürlicher Sprache* (KONVENS 2002), Saarbrücken, Germany.

[7] Martens, S. (2012) TüNDRA: TIGERSearch-style treebank querying as an XQuery-based web service. In: *Proc. of the joint CLARIN-D/DARIAH Workshop "Service-oriented Architectures (SOAs) for the Humanities: Solutions and Impacts"* (DH 2012), Hamburg, pp. 41-50.

[8] McGillivray, B., M. Passarotti and P. Ruffolo (2009) The Index Thomisticus Treebank Project: Annotation, Parsing and Valency Lexicon. *TAL* 50(2), pp. 103-127.

[9] Nardi, B. (1996) *Context and Consciousness: Activity Theory and Human-Computer Interaction*, MIT Press.

[10] Pajas, P. and J. Štěpánek (2009) System for Querying Syntactically Annotated Corpora. In: *Proc. of the ACL-IJCNLP 2009 Software Demonstrations*, Singapore, pp. 33-36.

[11] Rosén, V., K. De Smedt, P. Meurer, and Helge Dyvik (2012) An open infrastructure for advanced treebanking. In: *Proc. of the META-RESEARCH Workshop on Advanced Treebanking at LREC 2012*, Istanbul, pp. 22-29.

[12] Sugayama, K. and R. Hudson (2006) *Word Grammar: New Perspectives on a Theory of Language Structure* London: Continuum.

[13] Telljohann, H., E. Hinrichs, S, Kübler, H. Zinsmeister, and K. Beck (2012) *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*, Uni. Tübingen. (URL: http://www.sfs.uni-tuebingen.de/fileadmin/static/ascl/resources/tuebadz-stylebook-1201.pdf)

[14] Zeldes, A., J. Ritz, A. Lüdeling, and C. Chiarcos. (2009) ANNIS: A Search Tool for Multi-Layer Annotated Corpora. In: *Proceedings of Corpus Linguistics 2009*, Liverpool, UK.

Integration of Dependency Parsers for Bulgarian

Kiril Simov[†], Ginka Ivanova[†], Maria Mateva[‡], Petya Osenova[†]

[†]Linguistic Modelling Laboratory, IICT-BAS, Sofia, Bulgaria
[‡]Sofia University "St. Kl. Ohridski", Sofia, Bulgaria
kivs@bultreebank, givanova@bultreebank.org,
mateva.maria@gmail.com, petya@bultreebank.org

Abstract

Various parsing models — based on different parsing algorithms or different sets of features — produce errors in different places in the dependency trees — [6]. This observation initiated a wide range of research devoted to combining the outputs of various parsing models with the hope to achieve a better parsing result. In this paper we present our work on combining results from 14 parsing models for Bulgarian. First, we construct an extension of the Bulgarian treebank with the parses from each of the models. Then, we evaluate different combining approaches including three voting models and two machine learning approaches. Each of the weighting mechanisms is used by two different tree construction algorithms to find an optimal solution.

1 Introduction

The combination of several dependency parsers is motivated by the need of having an accurate parser even at the price of slower performance. Considering the outcomes of [10], we have performed several experiments to assemble the results of several parsing models using different types of voting and also exploiting machine learning techniques. We show that, contrary to the conclusion of [10], machine learning techniques can outperform voting techniques.

First, we tried fourteen different parsing models using two of the most popular parsing systems: MaltParser (see [7]) and MSTParser (see [5]). They were trained using different parsing algorithms and different sets of features. Our goal was to minimize the number of nodes for which there was no parsing model to infer the correct arc. For each of the fourteen models at our disposal, we constructed a version of the original treebank where we stored the arcs suggested by the parsing model. Thus, at the end we had fourteen additional treebanks — one per model. We used them for performing the voting experiments and for training a machine learning component to select the most appropriate arc for each node. We experimented with three voting approaches: (1) majority of the parsers that selected a

given arc; (2) maximum of the sum of parsers' accuracy values of the parsers that selected a given arc; and (3) average of the accuracy of the parsers that selected a given arc. The best result was achieved by the second approach. For machine learning approaches we performed two experiments: (1) selecting the correct arc among all arcs suggested by the fourteen models; and (2) ranking the arcs by comparing pairs of arcs. Here also the second approach was the best. In both cases we used two algorithms for constructing dependency trees from the trees suggested by the fourteen models: the incremental algorithm of [1] and the global algorithm for non-projective parsing of [5]. The most evident difference between the two models is shown by the second machine learning approach.

The structure of the paper is as follows: first, we discuss the related work; in the next section, the linguistic phenomena, encoded in the treebank, are presented; in Section 4 we introduce the parsing models that we have trained as well as the construction of an extended version of BulTreeBank to be used in the experiments; Section 5 describes the models that we exploit for the voting combination of parses; Section 6 presents the machine learning approaches relevant to the task; the last section concludes the paper.

2 Related work

Our work is inspired by the analysis of the two most influential dependency parsing models: transition-based and graph-based frameworks — [6]. They showed that these two frameworks made different errors on the same training and test datasets. The authors conclude the paper by proposing three approaches of using the advantages of both frameworks: (1) Ensemble systems — weighted combinations of both systems output; (2) Hybrid systems — design of a single system integrating the strengths of each framework; and (3) Novel approaches — based on combination of new training and inference methods. In their further work (see [8]) the authors present a hybrid system that combines the two models. In our work we consider option one — ensemble systems, since it is easier to implement and does not require further knowledge of the models and the design and implementation details of the parsing algorithms.

Another work devoted to ensemble systems is [10]. They tested different approaches to parser integration — different types of voting and meta-classification. The voting determines the correct arcs on the basis of majority of parsers that select given arcs. The weighted voting is using the accuracy of each parser in order to choose between them for each arc. Authors' conclusion is that meta-classification does not help for improving the result in comparison to voting. The authors divided the dependencies into two ways: majority dependencies and minority dependencies. Their conclusion is that meta-classification cannot provide a better selection on minority dependencies, and in this way it is comparable to voting. In our work we show that depending on the feature selections for the meta-classification we might outperform the voting approach. The results reported in [10] do not use a

special algorithm for the selection of dependencies. They do not require the result to be a tree. In our work we use two algorithms to ensure the construction of trees. We show that the improvement also depends on the algorithm for constructing the complete tree.

As mentioned above, we use two algorithms for construction of the dependency tree. The first algorithm is reported in [1]. It constructs the dependency tree incrementally starting from an empty tree and then selecting the arc with the highest weight that could extend the current partial tree. The algorithm decides for the best arc locally. We will denote this algorithm below as `LocTr`. The second algorithm is Chu-Liu-Edmonds algorithm for maximal spanning tree implemented in the MSTParser of [5]. This algorithm starts with a complete dependency graph including all possible dependency arcs. Then the algorithm selects the maximal spanning tree on the basis of the weights assigned to the potential arcs. In our work we use weights for the arcs assigned in two ways: (1) by voting; and (2) by metaclassifiers. The arcs that are not proposed by any of the parsers are deleted. This algorithm is global with respect to the selection of arcs. We will denote this algorithm below as `GloTr`.

3 The Linguistic Annotation of the Bulgarian Treebank (BulTreeBank)

BulTreeBank (see [9]) is a treebank that provides rich linguistic information going beyond the syntactic information. It has part-of-speech (POS) and full grammatical tags; lemmas; syntactic relations, based on Head-driven Phrase Structure Grammar (HPSG); named entities and co-references within a sentence.

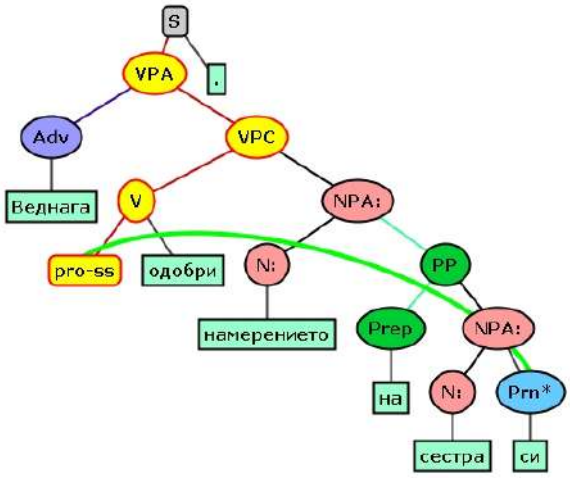


Figure 1: HPSG-based tree for the sentence "Веднага одобри намерението на сестра си" ('Immediately approved intention of sister his', He approved his sister's intention immediately).

Figure 1 presents the HPSG-based tree of a Bulgarian sentence. Here two things are worth mentioning: the presence of dependency information in the HPSG-based version (NPA means a nominal phrase of type head-adjunct) and the presence of a co-reference link between the unexpressed subject and the reflexive possessive pronoun. In the HPSG-based version of the treebank the unexpressed subject is represented explicitly only in the cases when it participates in a coreference chain as in this example. It is considered a property of the verb node, not a part of the constituent structure.

In Figure 2 the same sentence can be observed in a dependency format. The head-adjunct relation within the NPA nodes in the HPSG-based tree is projected into a head-modifier relation in the dependency tree. The arc labels are represented as ovals between word nodes. The coreference is represented as a secondary edge between the corresponding word nodes.

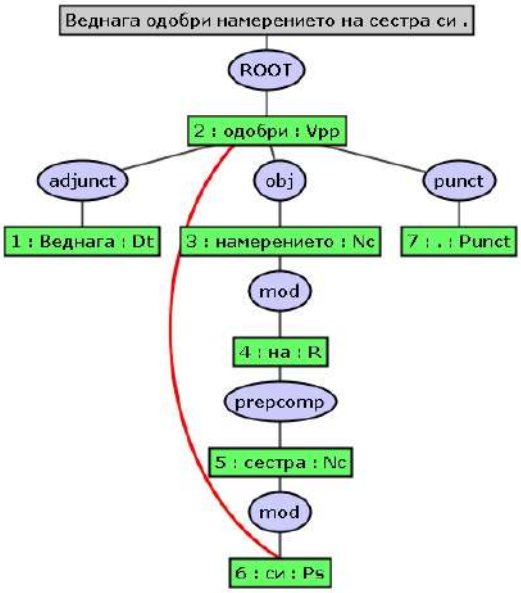


Figure 2: Dependency tree representation of the sentence from Figure 1.

In Table 1 the dependency tagset related to the Dependency part of the BulTreeBank is presented. In addition to the dependency tags we also have morphosyntactic tags attached to each word. For each lexical node the lemma is assigned. The number under the name of each relation indicates how many times the relation appears in the dependency version of BulTreeBank. Many parsers have been trained on data from BulTreeBank during the CoNLL 2006 shared task and after[1]. BulTreeBank appeared to be one of the best parsed treebanks — [3].

[1] See papers in the proceedings at http://aclweb.org/anthology/W/W06/#2900

Relation	Occurrences	Description
adjunct	12009	Adjunct (optional verbal argument)
clitic	2263	Short forms of the possessive pronouns
comp	18043	Complement (argument of non-finite verbs, copula, auxiliaries)
conj	6342	Conjunction in coordination
conjarg	7005	Argument (second, third, ...) of coordination
indobj	4232	Indirect Object
marked	2650	Marked (clause, introduced by a subordinator)
mod	42706	Modifier (dependants which modify nouns, adjectives, adverbs, ...)
obj	7248	Object (direct argument of a non-auxiliary verbal head)
pragadjunct	1612	Pragmatic adjunct
punct	28134	Punctuation
subj	14064	Subject
xadjunct	1826	Clausal adjunct
xcomp	4651	Clausal complement
xmod	2219	Clausal modifier
xprepcomp	168	Clausal complement of preposition
xsubj	504	Clausal subject

Table 1: Dependency relations and numbers of their occurrences in BulTreeBank.

4 Parser Models and Extension of the Treebank

For our experiments we have trained 12 models of MaltParser [7] using different parsing algorithms and different features, and MSTParser of [5] with two different sets of features. The models are: MaltParser and MSTParser.

Initially, we conducted a series of experiments with both parsers and BulTreeBank using 10-fold cross-validation. The original treebank on which the models were trained contains 11988 sentences. They were randomly divided in ten training and test sets in such a way that each sentence appeared in one test set. Each training set contains 10790 sentences and each test set — 1198. The number of tokens is 183649. For each of the models we performed ten training sessions. The trained model was applied on the corresponding test set for evaluation, but also the predicted parses for each sentence were stored. In this way, for each model we constructed a treebank of the suggested parses. Thus, at the end we had the original treebank plus fourteen treebanks of parses. We used all these treebanks for performing different combinations of parses from the different models.

For MSTParser we preselected the two best performing parsers on average with major difference in the scope of features (order). The first parser used features over a single edge, while the second one used features over pairs of adjacent edges. The rest of the parameters were chosen for the parser's optimal labeled and unlabeled accuracy, on average, of the experiments on BulTreeBank: MST01 model used the following features: loss-type: *punc*; second-order: *false*; training-iterations: *15*;

trainingk: *1*; decode-type: *non-proj*; create-forest: *true*; MST02 model used the following features: loss-type: *punc*; second-order: *true*; training-iterations: *15*; trainingk: *1*; decode-type: *non-proj*; create-forest: *true*.

For MaltParser we applied three Malt algorithms: Covington non-projective, Stack Eager and Stack Lazy. The predefined flow chart is set to learn. The actual configuration type of MaltParser is “singlemalt". The input data type is CoNLL-X shared task data format — see [3]. According to the Covington algorithm, each new token is attempted to be linked to the preceding token. In our study, we configured the Covington model with root and shift options set to true. During the parsing process, the root could be treated as a standard node and attached with a RightArc transition. The option Shift = true allows the parser to skip remaining tokens in Left (otherwise all tokens in Left must be analyzed before treating the next token). The Stack algorithms use a stack and a buffer, and produce a tree without post-processing by adding arcs between the two top nodes on the stack. Via a swap transition, we obtain non-projective dependency trees. The difference between the Eager algorithm and the Lazy algorithm is the time when the swap transition is applied (as soon as possible for the first algorithm and as late as possible respectively for the second one). The execution of algorithms with the LIBLINEAR method is faster than algorithms with the LIBSVM method and the results are better. On the basis of these six models we constructed additional six ones by extending the set of node features including more detailed description of grammatical features and lemma.

The results in terms of Labeled Attachment Scores (LAS) and Unlabeled Attachment Scores (UAS) for the fourteen models are given in Table 2.

	MLT01	**MLT02**	**MLT03**	**MLT04**	**MLT05**	**MLT06**	**MLT07**
LAS	0.842	0.788	0.843	0.809	0.825	0.820	0,863
UAS	0.886	0.835	0.887	0.860	0.869	0.869	0,900
	MLT08	**MLT09**	**MLT10**	**MLT11**	**MLT12**	**MST01**	**MST02**
LAS	0.810	0.872	0.844	0.851	0.847	0.852	0.828
UAS	0.849	0.909	0.886	0.887	0.888	0.898	0.872

Table 2: LAS and UAS results for the fourteen models.

The models do not predict the correct unlabeled arc for 1.99% of the words in the treebank. For the labeled arcs the percentage is 3.49%. Thus, the upper bound for the UAS measure is 98.01% and for the LAS measure is 96.51%. The results reported within this paper are still far from the upper bounds for both measures.

5 Combining Parses by Voting

We investigated three voting models: (1) the arcs are ranked by the number of the parsers that predicted them; (2) the arcs are ranked by the sum of the UAS measures for all parsers that predicted the arcs; and (3) the arcs are ranked by the average

of the UAS measures of the parsers that predicted the arcs. Let us consider an artificial example for arcs suggested by different models. They are given in Table 3

	MLT01	**MLT02**	**MLT03**	**MLT04**	**MLT05**	**MLT06**	**MLT07**
Node	Arc01	Arc02	Arc01	Arc01	Arc03	Arc02	Arc02
UAS	0.835	0.887	0.860	0.869	0.869	0.886	0,899
	MLT08	**MLT09**	**MLT10**	**MLT11**	**MLT12**	**MST01**	**MST02**
Node	Arc01	Arc04	Arc03	Arc03	Arc03	Arc02	Arc02
UAS	0.848	0.908	0.886	0.887	0.888	0.898	0.872

Table 3: Different arcs for a node in a tree. For orientation we also give UAS values for each model.

On the basis of these arc suggestions and UAS values for the different models we can calculate the three ranks. These ranks for voting are given in Table 4. As it can be seen, the different ranking models define a different ordering on the arcs. Different values also play significant role when the arcs are combined by the algorithms to form a dependency tree.

Arcs	**UASes**	**Rank01**	**Rank02**	**Rank03**
Arc01	0.835, 0.860, 0.869, 0,848	4	3.412	0.853
Arc02	0.887, 0.886, 0.899, 0.898, 0.872	5	4.442	0.888
Arc03	0.869, 0.886, 0.887, 0.888	4	3.530	0.883
Arc04	0.908	1	0.908	0.908

Table 4: Different ranking for voting.

We ran both algorithms (`LocTr` and `GloTr`) for construction of dependency trees on the basis of combinations of dependency parses over results from all models. Then following the suggestions of [10] we performed combinations by using the results from different models. Although our results are not drastically different, they show that combining only a few of the models could give better results than combining all the models. It is interesting that combining several parser models with best scores does not give the best result, but it is relevant to include at least one parser model with low score. Table 5 shows the results from combining all fourteen models and then the best combinations for 3, 4 and 5 models.

The results clearly show that ranking with average accuracy influences performance very negatively. The experiments demonstrate slight preference for the rank, based on the sum of accuracies for the various models. The good news is that several combinations achieved substantial improvement over the accuracy of the individual models.

6 Combining Parses by Machine Learning

Although [10] claimed that they could not implement a combination model that involves machine learning methods for selection of arcs and which improves sub-

Models	Algor.	Rank01 Number		Rank02 Sum		Rank03 Average	
		LAS	UAS	LAS	UAS	LAS	UAS
All	LocTr	0.856	0.919	0.857	0.921	0.788	0.843
	GloTr	0.883	0.919	0.885	0.921	0.804	0.836
MLT08,	LocTr	0.876	0.920	0.878	0.922	0.844	0.885
MLT09, MLT11	GloTr	0.869	0.903	0.875	0.909	0.858	0.893
MLT07, MLT08,	LocTr	0.872	0.918	0.877	0.922	0.830	0.871
MLT09, MLT11	GloTr	0.873	0.907	0.882	0.916	0.852	0.885
MLT07, MLT08,	LocTr	0.875	0.923	0.875	**0.924**	0.828	0.872
MLT09, MLT11, MST01	GloTr	0.886	0.918	0.888	**0.921**	0.850	0.881
MLT07, MLT09,	LocTr	0.874	0.923	0.875	**0.924**	0.828	0.872
MLT12, MST01, MST02	GloTr	0.888	0.923	0.891	**0.925**	0.840	0.872
MLT01, MLT07,	LocTr	0.872	**0.925**	0.873	**0.925**	0.825	0.872
MLT09, MLT11,	GloTr	0.891	**0.925**	0.892	**0.925**	0.838	0.869
MLT12, MST01, MST02							

Table 5: Voting using algorithms LocTr and GloTr for tree construction.

stantially over the voting approaches, we performed several experiments in order to test different feature sets. Our goal was to test this claim on our data. Our intuition was that independently from the task: direct dependency parsing or combining existing parses, we need machine learning mechanisms that are able to learn also hard-to-predict dependencies.

Here we conducted two experiments using the extended treebank for training and testing of machine learning techniques for ranking suggested arcs. Again, as in the case of voting, we tested both algorithms for the construction of the dependency trees.

The experiments were conducted using the package RandomForest[2] of the system R[3]. Random Forest (see [2]) constructs randomly several decision trees over the data and then assembles their predictions. We chose this package because of the following characteristics of RandomForest: (1) it supports classification and regression methods; and (2) it does not overfit. The data for RandomForest is prepared as a vector of feature values where one of the features determines what the package predicts.

The treebank was divided again into training and test parts in the same proportion: 90% for training and 10% for test. The division was made orthogonal to the division used for the creation of the fourteen treebanks used to train the models. In this way, we reduced the bias from the training of the parsing models.

Our goal was to evaluate each arc suggested by a parsing model as a correct arc for a given context or as an incorrect arc for the context. Each arc was modeled by three features: relation (Rel), distance to the parent node measured as a number

[2]http://cran.r-project.org/web/packages/randomForest/randomForest.pdf

[3]http://www.r-project.org/

of words in the sentence between the two nodes (Dist) and direction of the parent node (Dir). Direction could be Left, which means the parent node is on the left side of the node, and Right — the parent node is on the right side of the node. In cases when these features are not relevant, we simply use Arc or ArcN for the arc features. The features for each word node include: the word form (WF), lemma (Lemma), morphosyntactic tag (POS). Again, when the features are not important in isolation, we denote them as Word with optional modification.

For the first experiments we have evaluate the appropriateness of one arc between dependant node Word and a head (or parent) node. In this experiment we have used all the arcs for a given node as a context together with the trigrams around the node, and the parent node. The node has features Word, the parent ParentWord. A representation of this data as a value vector for RandomForest is presented in Table 6.

Feature	**Value**
Word	the current node
WordBefore	the word before the current node
WordAfter	the word after the current node
Arc01, ..., ARC14	arcs suggested by each of the fourteen models
ParentWord	the parent word
PWordBefore	the word before the parent word
PWordAfter	the word after the parent word
EvaluationArc	one of the arcs suggested by one of the models for the node
CorrectIncorrect	true or false depending on whether the EvaluationArc arc is the correct one for the node

Table 6: Feature vector used with RandomForest for the first experiment.

All tuples for all arcs in the training and the test parts of the extended treebank were generated. The tuples generated from the training part of the treebank were used to train the RandomForest, then the model was applied on test set to classify each of the tuples as correct or incorrect.

The first experiments were done using the classification method of RandomForest. The results were very disappointing. Table 7 presents the results for this classification method. These are the results for the tuples, not the dependencies.

	Correct	**Incorrect**
True	65.2%	73.4%
False	34.8%	26.6%

Table 7: Results from the classification of tuples.

The results obtained for tree construction were very bad. Our explanation of the results is the fact that we had only two classes. Thus the classifier was sensitive to values close to 50%. Thus, dependencies that were hard to predict correctly very often received weights near to this threshold, and small fluctuations above or under

it caused wrong classification. The first attempt to avoid this problem was to divide the training and the test sets according to the POS information for the wordform. Unfortunately, this did not improve the situation substantially. The second attempt we performed was to switch from this classification method to regression.

Model	Algorithm	LAS	UAS
MLearn14	LocTr	0.859	0.920
MLearn14	GloTr	0.896	**0.925**

Table 8: Results for the algorithms LocTr and GloTr when the correctness of an arc is evaluated with respect to all 14 models.

In order to use the regression method, we first encoded all the symbolic data as numerical data. The results from regression are between 8 and 17 mean of squared residuals which are much better than the results for classification. Again, the training and test sets were divided according to the POS tag of Word. Each element of the test set received a weight returned from the regression model for the corresponding part of speech. These weights were used by the algorithms LocTr and GloTr to construct the dependency trees. The results from the two algorithms are presented in Table 8.

These results confirm the conclusions of [10] that machine learning hardly improves over the voting methods. Our explanation of the results is that the machine learning component is learning to do voting selecting the best suggestion of the fourteen parsing models. Thus, using all the fourteen arcs as a context is not a good idea. In order to avoid such cases, we designed a second experiment in which the candidate arc was evaluated in the context of one alternative arc for a given node. In this way, we were trying to implement a model which learns which arc from two candidate arcs is better for a node. Thus, for the second experiment we have evaluated the appropriateness of one arc between a dependant node Word and a head (or parent) node in context of one alternative arc for the same node with the trigrams around the node, and the parent node. The node has features Word, the parent has features ParentWord. For the alternative arc we also have used as a context the grammatical features of its parent node — AltParentPOS. A representation of this data as a value vector for RandomForest is presented in Table 9.

The training and test sets were divided further according to the POS tag of Word. The mean of squared residuals for the different parts of speech are between 4 and 12. Each element of the test set received a weight returned from the regression model. But this time some of the arcs in the treebank could receive more than one weight because each arc could have more than one alternative arc. We used these weights to define three models: (1) prefer as a weight for the arc the maximum of the weights for the tuples for the arc; (2) prefer as a weight for the arc the minimum of the weights for the tuples for the arc; and (3) assign as a weight for the arc the multiplication of the weights for the tuples for the arc. For all three models we ran the algorithms LocTr and GloTr to construct the dependency trees. The results from this experiment are presented in Table 10.

Feature	Value
Word	the current node
WordBefore	the word before the current node
WordAfter	the word after the current node
ArcAlt	alternative arc
AltParentPOS	grammatical features for alternative parent node
ParentWord	the parent word
PWordBefore	the word before the parent word
PWordAfter	the word after the parent word
EvaluationArc	the arc which we compare with the alternative arc ArcAlt
CorrectIncorrect	true or false depending on whether the EvaluationArc arc is the correct one for the node

Table 9: Feature vector used for the second experiment with RandomForest.

Model	Algorithm	MinRank		MaxRank		MultRank	
		LAS	UAS	LAS	UAS	LAS	UAS
MLearn01	LocTr	0.844	0.903	0.843	0.902	0.828	0.887
MLearn01	GloTr	0.899	**0.929**	0.897	0.926	0.886	0.915

Table 10: Results for the algorithms LocTr and GloTr when the correctness of an arc is evaluated with respect to one alternative arc.

These results show that machine learning could improve substantially the voting models. In our case the improvement is near half percent. These results show better the difference between the two combining algorithms (LocTr and GloTr) — the results from the global optimization are better than the local optimization.

7 Conclusion and Future Work

In this paper we presented several approaches for combining parses produced by several parsing models. These approaches include three types of voting and two machine learning approaches. Also, for the construction of the combined trees we used two different algorithms — one performing local optimization and one performing global optimization. The better ranking of the suggested arcs also demonstrates the better performance of the global algorithm.

Although we achieved some substantial improvement over the individual models (near 3% improvement), the results are far from the potential maximum UAS measure of 98.01% and for LAS measure — of 96.51%. In spite of that, our best result is near to the state-of-the-art performance for Bulgarian — 93.5 % — see [4]. Thus, our first goal to have access to a better parser for Bulgarian is achieved.

In future, we would like to combine better individual parsing models. In these experiments we executed the very basic parser configurations without extensive optimization. We hope that starting from better models, the combined result will be also better. Also, we would like to include more sentences in the original treebank

in order to have better coverage of the problematic cases. Last, but not least, it will be interesting to see the impact of adding semantic features to the model.

Acknowledgements

This research has received partial funding from the EC's FP7 (FP7/2007-2013) under grant agreement number 611760. We would like to thank Laura Tolo̧si for helping us with R system.

References

[1] Giuseppe Attardi and Felice Dell'Orletta. 2009. Reverse revision and linear tree combination for dependency parsing. In *Proceedings of NAACL HLT 2009, Companion Volume: Short Papers*, pages 261–264.

[2] Leo Breiman. Random forests. 2001. *Machine Learning*, 45(1):5–32, 2001.

[3] Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164.

[4] Andre Martins, Noah Smith, Mario Figueiredo, and Pedro Aguiar. 2011. Dual decomposition with many overlapping components. In *Proceedings of EMNLP 2011*, pages 238–249.

[5] Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. PhD thesis.

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

[7] Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: a data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*.

[8] Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958.

[9] Kiril Simov, Petya Osenova, Alexander Simov, and Milen Kouylekov. 2004. Design and implementation of the Bulgarian HPSG-based treebank. *Research on Language & Computation*, 2(4):495–522, 2004.

[10] Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of NAACL-2010*, pages 649–652.