Informed ways of improving data-driven dependency parsing for German

Wolfgang Seeker
University of Stuttgart
Inst. für Maschinelle Sprachverarbeitung
seeker@ims.uni-stuttgart.de

Bernd Bohnet
University of Stuttgart
Inst. für Maschinelle Sprachverarbeitung
Bernd.Bohnet@ims.uni-stuttgart.de

Lilja Øvrelid
University of Potsdam
Institut für Linguistik
ovrelid@uni-potsdam.de

Jonas Kuhn
University of Stuttgart
Inst. für Maschinelle Sprachverarbeitung
jonas@ims.uni-stuttgart.de

Abstract

We investigate a series of targeted modifications to a data-driven dependency parser of German and show that these can be highly effective even for a relatively well studied language like German if they are made on a (linguistically and methodologically) informed basis and with a parser implementation that allows for fast and robust training and application. Making relatively small changes to a range of very different system components, we were able to increase labeled accuracy on a standard test set (from the CoNLL 2009 shared task), ignoring gold standard part-of-speech tags, from 87.64% to 89.40%. The study was conducted in less than five weeks and as a secondary project of all four authors. Effective modifications include the quality and combination of auto-assigned morphosyntactic features entering machine learning, the internal feature handling as well as the inclusion of global constraints and a combination of different parsing strategies.

1 Introduction

The past years have seen an enormous surge of interest in dependency parsing, mainly in the data-driven paradigm, and with a particular emphasis on covering a whole set of languages with a single approach. The reasons for this interest are manifold; the availability of shared task data from various CoNLL conferences (among others (Buchholz and Marsi, 2006; Hajič et al., 2009)), comprising collections of languages based on a single representation format, has certainly been instrumental. But likewise, the straightforward usefulness of dependency representations for a number of tasks plays an important role. The relative language independence of the representations makes dependency parsing particularly attractive for multilingually oriented work, including machine translation.

As data-driven approaches to dependency parsing have reached a certain level of maturity, it may appear as if further improvements of parsing performance have to rely on relatively advanced tuning procedures, such as sophisticated automatic feature selection procedures or combinations of different parsing approaches with complementary strengths. It is indeed still hard to pinpoint the structural properties of a language (or annotation scheme) that make the parsing task easier for a particular approach, so it may seem best to leave the decision to a higher-level procedure.

This paper starts from the suspicion that while sophisticated tuning procedures are certainly helpful, one should not underestimate the potential of relatively simple modifications of the experimental set-up, such as a restructuring of aspects of the dependency format, a targeted improvement of the quality of automatically assigned features, or a simplification of the feature space for machine learning – the modifications just have to be made in an informed way. This presupposes two things: (i) a thorough linguistic understanding of the issues at hand, and (ii) a relatively powerful and robust experimental machinery which allows for experimentation in various directions and which should ideally support a fast turn-around cycle.

We report on a small pilot study exploring the potential of relatively small, informed modifications as a way of improving parsing accuracy even for a language that has received considerable attention in the parsing literature, including the dependency parsing literature, namely German. Within a timeframe of five weeks and spending only a few hours a day on the project (between a group of four people), we were able to reach some surprising improvements in parsing accuracy.

By way of example, we experimented with modifications in a number of rather different system areas, which we will discuss in the course of this paper after a brief discussion of related work and the data basis in Section 2. Based on a second-order maximum spanning tree algorithm, we used a hash kernel to facilitate the mapping of the features onto their weights for a very large number of features (Section 3); we modified the dependency tree representation for prepositional phrases, adding hierarchical structure that facilitates the picking up of generalizations (Section 4). We take advantage of a morphological analyzer to train an improved part-of-speech tagger (Section 5), and we use knowledge about the structure of morphological paradigms and the morphology-syntax interface in the feature design for machine learning (Section 6). As is known from other studies, the combination of different parsing strategies is advantageous; we include a relatively simple parser stacking procedure in our pilot study (Section 7), and finally, we apply Integer Linear Programming in a targeted way to add some global constraints on possible combinations of arc labels with a single head (Section 8). Section 9 offers a brief conclusion.

2 Related Work and Data Basis

We quickly review the situation in data-driven dependency parsing in general and on applying it to German specifically.

The two main approaches to data-driven dependency parsing are transition based dependency parsing (Nivre, 2003; Yamada and Matsumoto, 2003; Titov and Henderson, 2007) and maximum spanning tree based dependency parsing (Eisner, 1996; Eisner, 2000; McDonald and Pereira, 2006). Transition based parsers typically have a linear or quadratic complexity (Attardi, 2006). Nivre (2009) introduced a transition based non-projective parsing algorithm that has a worst case quadratic complexity and an expected linear parsing time. Titov and Henderson (2007) combined a transition based parsing algorithm, using beam search, with a latent variable machine learning technique.

Maximum spanning tree based dependency parsers decompose a dependency structure into *factors*. The factors of the first order maximum spanning tree parsing algorithm are edges consisting of the head, the dependent (child) and the edge label. This algorithm has a quadratic complexity. The second order parsing algorithm of McDonald and Pereira (2006) uses a separate algorithm for edge labeling. In addition to the first order factors, this algorithm uses the edges to those children which are closest to the dependent and has a complexity of O(n^3). The second order algorithm of Carreras (2007) uses in addition to McDonald and Pereira (2006) the child of the dependent occurring in the sentence between the head and the dependent as well as the edge from the dependents to a grandchild. The edge labeling is an integral part of the algorithm which requires an additional loop over the labels. This algorithm therefore has a complexity of O(n^4). Johansson and Nugues (2008) reduced the required number of loops over the edge labels by considering only the edges that existed in the training corpus for a distinct head and child part-of-speech tag combination.

Predating the surge of interest in data-based dependency parsing, there is a relatively long tradition of dependency parsing work on German, including for instance Menzel and Schröder (1998) and Duchier and Debusmann (2001). German was included in the CoNLL shared tasks in 2006 (Multilingual Dependency Parsing, (Buchholz and Marsi, 2006)) and in 2009 (Syntactic and Semantic Dependencies in Multiple Languages, (Hajič et al., 2009)) with data based on the TIGER corpus (Brants et al., 2002) in both cases. Since the original TIGER treebank is in a hybrid phrase-structural/dependency format with a relatively flat hierarchical structure, conversion to a pure dependency format involves some non-trivial steps. The 2008 ACL Workshop on Parsing German included a specific shared task on dependency parsing of German (Kübler, 2008), based on two sets of data: again the TIGER corpus – however with a different conversion routine than for the CoNLL tasks – and the TüBa-D/Z corpus (Hinrichs et al., 2004).

In the 2006 CoNLL task and in the 2008 ACL Workshop task, the task was dependency parsing with given gold standard part-of-speech tags from the corpus. This is a valid way of isolating the specific subproblem of parsing, however it is clear that the task does not reflect the application setting which includes noise from automatic part-of-speech tagging. In the 2009 CoNLL task, both gold standard tags and automatically assigned tags were provided. The auto-tagged version was created with the standard model of the TreeTagger (Schmid, 1995) (i.e., with no domain-specific tagger training).

In our experiments, we used the data set from the 2009 CoNLL task, for which the broadest comparison of recent parsing approaches exists. The highest-scoring system in the shared task was Bohnet (2009) with a labeled accuracy (LAS) of 87.48%, on auto-tagged data. The highest-scoring (in fact the only) system in the dependency parsing track of the 2008 ACL Workshop on parsing German was Hall and Nivre (2008) with an LAS of 90.80% on gold-tagged data, and with a data set that is not comparable to the CoNLL data.[1]

3 Hash Kernel

Our parser is based on a second order maximum spanning tree algorithm and uses MIRA (Crammer et al., 2006) as learning technique in combination with a hash kernel. The hash kernel has a higher accuracy since it can use additional features found during the creation of the dependency tree in addition to the features extracted from the training examples. The modification to MIRA is simple: we replace the feature-index mapping that maps the features to indices of the weight vector by a random function. Usually, the feature-index mapping in the support vector machine has two tasks: The mapping maps the features to an index and it filters out features that never occurred in a dependency tree. In our approach, we do not filter out these features, but use them as additional features. It turns out that this choice improves parsing quality. Instead of the feature-index mapping we use the following hash function:[2]

$h \leftarrow |(l\ xor(l \vee \text{0xffffffff00000000} >> 32))\%\ \text{size}|$

The Hash Kernel for structured data uses the hash function $h : J \to \{1...n\}$ to index ϕ where ϕ maps the observations X to a feature space. We define $\phi(x, y)$ as the numeric feature representation indexed by J. The learning problem is to fit the function F so that the errors of the predicted parse tree y are as low as possible. The scoring function of the Hash Kernel is defined as:[3]

$$F(x, y) = \overrightarrow{w} * \overline{\phi}(x, y)$$

For different j, the hash function $h(j)$ might generate the same value k. This means that the hash function maps more than one feature to the same weight which causes weight collisions. This procedure is similar to randomization of weights (features), which aims to save space by sharing values in the weight vector (Blum, 2006; Rahimi and Recht, 2008). The Hash Kernel shares values when collisions occur that can be considered as an approximation of the kernel function, because a weight might be adapted due to more than one feature. The approximation works very well with a weight vector size of 115 million values.

With the Hash Kernel, we were able to improve on a baseline parser that already reaches a quite high LAS of 87.64% which is higher than the top score for German (87.48%) in the CoNLL Shared task 2009. The Hash Kernel improved that value by 0.42 percentage points to 88.06%. In addition to that, we obtain a large speed up in terms of parsing time. The baseline parser spends an average of 426 milliseconds to parse a sentence of the test

[1]To get an idea of how the data sets compare, we trained the version of our parser described in Section 3 (i.e., without most of the linguistically informed improvements) on this data, achieving labeled accuracy of 92.41%, compared to 88.06% for the 2009 CoNLL task version.

[2]$>> n$ shifts n bits right, and % is the modulo operation.
[3]$\overrightarrow{w}$ is the weight vector and the size of $\overrightarrow{w}$ is n.

set and the parser with Hash Kernel only takes 126 milliseconds which is an increase in speed of 3.4 times. We get the large speed up because the memory access to a large array causes many CPU cache misses which we avoid by replacing the feature-index mapping with a hash function. As mentioned above, the speedup influences the experimenters' opportunities for explorative development since it reduces the turnaround time for experimental trials.

4 Restructuring of PPs

In a first step, we applied a treebank transformation to our data set in order to ease the learning for the parser. We concentrated on prepositional phrases (PP) to get an idea how much this kind of transformation can actually help a parser. PPs are notoriously flat in the TIGER Treebank annotation (from which our data are derived) and they do not embed a noun phrase (NP) but rather attach all parts of the noun phrase directly at PP level. This annotation was kept in the dependency version and it can cause problems for the parser since there are two different ways of annotating NPs: (i) for normal NPs where all dependents of the noun are attached as daughters of the head noun and (ii) for NPs in PPs where all dependents of the noun are attached as daughters to the preposition thus being sisters to their head noun. We changed the annotation of PPs by identifying the head noun in the PP and attaching all of its siblings to it. To find the correct head, we used a heuristic in the style of Magerman (1995). The head is chosen by taking the rightmost daughter of the preposition that has a category label according to the heuristic and is labeled with NK (noun kernel element).

Table 1 shows the parser performance on the data after PP-restructuring.[4] The explanation for the benefit of the restructuring is of course that now there is only one type of NP in the whole corpus which eases the parser's task to correctly learn and identify them.

	dev. set		test set	
	LAS	UAS	LAS	UAS
hash kernel	87.40	89.79	88.06	90.24
+restructured	87.49	89.97	88.30	90.44

Table 1: Parser performance on restructured data

Since restructuring parts of the corpus seems beneficial, there might be other structures where more consistent annotation could help the parser, e. g., coordination or punctuation (like in the 2008 ACL Workshop data set, cp. Footnote 1).

5 Part-of-Speech Tagging

High quality part-of-speech (PoS) tags can greatly improve parsing quality. Having a verb wrongly analyzed as a noun and similar mistakes are very likely to mislead the parser in its decision process. A lot of the parser's features include PoS tags and reducing the amount of errors during PoS tagging will therefore reduce misleading feature values as well. Since the quality of the automatically assigned PoS tags in the German CoNLL '09 data is not state-of-the-art (see Table 2 below), we decided to retag the data with our own tagger which uses additional information from a symbolic morphological analyzer to direct a statistical classifier.

For the assignment of PoS tags, we apply a standard maximum entropy classification approach (see Ratnaparkhi (1996)). The classes of the classifier are the PoS categories defined in the *Stuttgart-Tübingen Tag Set (STTS)* (Schiller et al., 1999). We use standard binarized features like the word itself, its last three letters, whether the word is capitalized, contains a hyphen, a digit or whether it consists of digits only. As the only non-binary feature, word length is recorded. These standard features are augmented by a number of binary features that support the classification process by providing a preselection of possible PoS tags. Every word is analyzed by DMOR, a finite state morphological analyzer, from whose output analyses all different PoS tags are collected and added to the feature set. For example, DMOR assigns the PoS tags NN (common noun) and ADJD (predicative adjective) to the word *gegan-*

[4]Note that we are evaluating against a gold standard here (and in the rest of the paper) which has been restructured as well. With a different gold standard one could argue that the absolute figures we obtain are not fully comparable with the original CoNLL shared task. However, since we are doing dependency parsing, the transformation does neither add nor remove any nodes from the structure nor do we change any labels. The only thing that is done during the transformation is the reattachment of some daughters of a PP. This is only a small modification, and it is certainly linguistically warranted.

gen (gone). From these analyses two features are generated, namely *possible-tag:NN* and *possible-tag:ADJD*, which are strong indicators for the classifier that one of these classes is very likely to be the correct one. The main idea here is to use the morphological analyzer as a sort of lexicon that preselects the set of possible tags beforehand and then use the classifier to do the disambiguation (see Jurish (2003) for a more sophisticated system based on Hidden-Markov models that uses roughly the same idea). Since the PoS tags are included in the feature set, the classifier is still able to assign every class defined in *STTS* even if it is not in the preselection. Where the morphological analyzer does not know the word in question we add features for every PoS tag representing a productive word class in German, making the reasonable assumption that the morphology knows about all closed-class words and word forms. Finally, we add word form and possible tag features for the previous and the following word to the feature set thus simulating a trigram tagger. We used the method of Kazama and Tsujii (2005) which uses inequality constraints to do a very efficient feature selection[5] to train the maximum entropy model.

We annotated the entire corpus with versions of our own tagger, i.e., the training, development and test data. In order to achieve a realistic behavior (including remaining tagging errors, which the parser may be able to react to if they are systematic), it was important that each section was tagged without any knowledge of the gold standard tags. For the development and test portion, this is straightforward: we trained a model on the gold PoS of the training portion of the data and applied it to retag these two portions. Retagging the training portion was a bit trickier since we could not use a model trained on the same data, but at the same time, we wanted to use a tagger of similarly high quality – i.e. one that has seen a similar amount of training data. The training set was therefore split into 20 different parts and for every split, a tagging model was trained on the other 19 parts which then was used to retag the remaining 20th part. Table 2 shows the quality of our tagger evaluated on the German CoNLL '09 data in terms of accuracy and compares it to the originally annotated PoS tags which have been assigned by using the TreeTagger (Schmid, 1995) together with the German tagging model provided from the TreeTagger website. Tagging accuracy improves consistently by about 2 percentage points which equates to an error reduction of 44.55 % to 49.0 %.

	training	development	test
original	95.69	95.51	95.46
retagged	97.61	97.71	97.52
error red.	44.55%	49.00%	45.37%

Table 2: Tagging accuracy

Table 3 shows the parser performance when trained on the newly tagged data. The considerable improvements in tagging accuracy visibly affect parsing accuracy, raising both the labeled and the unlabeled attachment score by 0.66 percentage points (LAS) and 0.51 points (UAS) for the development set and by 0.45 points (LAS) and 0.64 points (UAS) for the test set.

	dev. set		test set	
	LAS	UAS	LAS	UAS
restructured	87.49	89.97	88.30	90.44
+retagged	88.15	90.48	88.75	91.08

Table 3: Parser performance on retagged data

6 Morphological Information

German, as opposed to English, exhibits a relatively rich morphology. Predicate arguments and nominal adjuncts are marked with special case morphology which allows for a less restricted word order in German. The German case system comprises four different case values, namely nominative, accusative, dative and genitive case. Subjects and nominal predicates are usually marked with nominative case, objects receive accusative or dative case and genitive case is usually used to mark possessors in possessive constructions. There are also some temporal and spatial nominal adjuncts which require certain case values. Since case is used to mark the function of a noun phrase in a clause, providing case information to a parser might improve its performance.

The morphological information in the German CoNLL '09 data contains much more information than case alone and previous models (baseline,

[5]We used a width factor of 1.0.

hash kernel, retagged) have used all of it. However, since we aim to improve a syntactic parser, we would like to exclude all morphological information from the parsing process that is not obviously relevant to syntax, e. g. mood or tense. By reducing the morphological annotations to those that are syntactically relevant, we hope to reduce the noise that is introduced by irrelevant information. (One might expect that machine learning and feature selection should "filter out" irrelevant features, but given the relative sparsity of unambiguous instances of the linguistically relevant effects, drawing the line based on just a few thousand sentences of positive evidence would be extremely hard even for a linguist.)

We annotated every case-bearing word in the corpus with its case information using DMOR. With case-bearing words, we mean nouns, proper nouns, attributive adjectives, determiners and all kinds of pronouns. Other types of morphological information was discarded. We did not use the manually annotated and disambiguated morphological information already present in the corpus for two reasons: the first one is the same as with the PoS tagging. Since it is unrealistic to have gold-standard annotation in a real-world application which deals with unseen data, we want the parser to learn from and hopefully adapt to imperfectly annotated data. The second reason is the German-inherent form syncretism in nominal paradigms. The German noun inflection system is with over ten different (productive and non-productive) inflectional patterns quite complicated, and to make matters worse, there are only five different morphological markers to distinguish 16 different positions in the pronoun, determiner and adjective paradigms and eight different positions in the noun paradigms. Some positions in the paradigm will therefore always be marked in the same way and we would like the parser to learn that some word forms will always be ambiguous with respect to their case value.

We also conducted experiments where we annotated number and gender values in addition to case. The idea behind this is that number and gender might help to further disambiguate case values. The downside of this is the increase in feature values. Combining case and number features means a multiplication of their values creating eight new feature values instead of four. Adding gender annotation raises this number to 24. Beside the disambiguation of case, there is also another reason why we might want to add number and gender: Inside a German noun phrase, all parts have to agree on their case and number feature in order to produce a well-formed noun phrase. Furthermore, the head noun governs the gender feature of the other parts. Thus, all three features can be relevant to the construction of a syntactic structure.[6] Table 4 shows the results of our experiments with morphological features.

	dev. set		test set	
	LAS	UAS	LAS	UAS
retagged	88.15	90.48	88.75	91.08
no morph.	87.78	90.18	88.60	90.92
+case	88.04	90.48	88.77	91.13
+c+n	88.21	90.62	88.88	91.13
+c+n+g	87.96	90.33	88.73	90.99

Table 4: Parser performance with morph. information (c=case, n=number, g=gender)

The *no morph* row in Table 4 shows, that using no morphological information at all decreases parser performance. When only case values are annotated, the parser performance does not change much in comparison to the retagged model, so there is no benefit here. Adding number features on the other hand improves parsing results significantly. This seems to support our intuition that number helps in disambiguating case values. However, adding gender information does not further increase this effect but hurts parser performance even more than case annotation alone. This leaves us with a puzzle here. Annotating case and number helps the parser, but case alone or having case, number and gender together affects performance negatively. A possible explanation might be that the effect of the gender information is masked by the increased number of feature values (24) which confuses the parsing algorithm.

7 Parser Stacking

Nivre and McDonald (2008) show how two different approaches to data-driven dependency pars-

[6]Person would be another syntactically relevant information. However, since we are dealing with a newspaper corpus, first and second person features appear very rarely.

ing, the graph-based and transition-based approaches, may be combined and subsequently learn to complement each other to achieve improved parsing results for different languages.

MaltParser (Nivre et al., 2006) is a language-independent system for data-driven dependency parsing which is freely available.[7] It is based on a deterministic parsing strategy in combination with treebank-induced classifiers for predicting parsing actions. MaltParser employs a rich feature representation in order to guide parsing. For the training of the Malt parser model that we use in the stacking experiments, we use learner and parser settings identical to the ones optimized for German in the CoNLL-X shared task (Nivre et al., 2006). Furthermore, we employ the technique of pseudo-projective parsing described in Nilsson and Nivre (2005) and a split prediction strategy for predicting parse transitions and arc labels (Nivre and Hall, 2008).[8] In order to obtain automatic parses for the whole data set, we perform a 10-fold split. For the parser stacking, we follow the approach of Nivre and McDonald (2008), using MaltParser as a guide for the MST parser with the hash kernel, i.e., providing the arcs and labels assigned by MaltParser as features. Table 5 shows the scores we obtain by parser stacking. Although our version of MaltParser does not quite have the same performance as for instance the version of Hall and Nivre (2008), its guidance leads to a small improvement in the overall parsing results.

	dev. set		test set	
	LAS	UAS	LAS	UAS
MaltParser	82.47	85.78	83.84	86.8
our parser	88.21	90.62	88.88	91.13
+stacking	88.42	90.77	89.28	91.40

Table 5: Stacked parser performance with guidance by MaltParser

[7]http://maltparser.org

[8]The feature models make use of information about the lexical form (FORM), the predicted PoS (PPOS) and the dependency relation constructed thus far during parsing (DEP). In addition, we make use of the predicted values for other morphological features (PFEATS). We employ the arc-eager algorithm (Nivre, 2003) in combination with SVM learners, using LIBSVM with a polynomial kernel.

8 Relabeling

In the relabeling step, we pursue the idea that some erroneous parser decisions concerning the distribution of certain labels might be detected and repaired in post-processing. In German and in most other languages, there are syntactic restrictions on the number of subjects and objects that a verb might select. The parser will learn this behavior during training. However, since it is using a statistical model with a limited context, it can still happen that two or more of the same grammatical functions are annotated for the same verb. But having two subjects annotated for a single verb makes this particular clause uninterpretable for subsequently applied tasks. Therefore, we would like to detect those doubly annotated grammatical functions and correct them in a controlled way.

The detection algorithm is simple: Running over the words of the output parse, we check for every word whether it has two or more daughters annotated with the same grammatical function and if we find one, we relabel all of its daughters.[9] For the relabeling, we applied a dependency-version of the function labeler described in Seeker et al. (2010) which uses a maximum entropy classifier that is restrained by a number of hard constraints implemented as an Integer Linear Program. These constraints model the aforementioned selectional restrictions on the number of certain types of verbal arguments. Since these are hard constraints, the labeler is not able to annotate more than one of those grammatical functions per verb. If we count the number of sentences that contain doubly annotated grammatical functions in the best parsing results from the previous section, we get 189 for the development set and 153 for the test set. About two thirds of the doubly annotated functions are subjects and the biggest part of the remaining third are accusative objects which are the most common arguments of German verbs.

Table 6 shows the final results after relabeling the output of the best performing parser configuration from the previous section. The improvements on the overall scores are quite small, which

[9]The grammatical functions we are looking for are SB (subject), OA (accusative object), DA (dative), OG (genitive object), OP (prepositional object), OC (clausal object), PD (predicate) and OA2 (second accusative object).

	dev. set		test set	
	LAS	UAS	LAS	UAS
stacking	88.42	90.77	89.28	91.40
+relabeling	88.48	90.77	89.40	91.40

Table 6: Parse quality after relabeling

is partly due to the fact that the relabeling affects only a small subset of all labels used in the data. Furthermore, the relabeling only takes place if a doubly annotated function is detected; and even if the relabeling is applied we have no guarantee that the labeler will assign the labels correctly (although we are guaranteed to not get double functions). Table 7 shows the differences in precision and recall for the grammatical functions between the original and the relabeled test set. As one can see, scores stay mostly the same except for SB, OA and DA. For OA, scores improve both in recall and precision. For DA, we trade a small decrease in precision for a huge improvement in recall and vice versa for SB, but on a much smaller scale. Generally spoken, relabeling is a local repair strategy that does not have so much effect on the overall score but can help to get some important labels correct even if the parser made the wrong decision. Note that the relabeler can only repair incorrect label decisions, it cannot help with wrongly attached words.

	original		relabeled	
	rec	prec	rec	prec
DA	64.2	83.2	74.7	79.6
OA	88.9	85.8	90.7	88.2
OA2	0.0	NaN	0.0	NaN
OC	95.2	93.5	95.1	93.7
OG	33.3	66.7	66.7	80.0
OP	54.2	80.8	54.2	79.9
PD	77.1	76.8	77.1	76.8
SB	91.0	90.6	90.7	93.7

Table 7: Improvements on grammatical functions in the relabeled test set

9 Conclusion

We presented a sequence of modifications to a data-driven dependency parser of German, departing from a state-of-the-art set-up in an implementation that allows for fast and robust training and application. Our pilot study tested what can be achieved in a few weeks if the data-driven technique is combined with a linguistically informed approach, i.e., testing hypotheses of what should be particularly effective in a very targeted way. Most modifications were relatively small, addressing very different dimensions in the system, such as the handling of features in the Machine Learning, the quality and combination of automatically assigned features and the ability to take into account global constraints, as well as the combination of different parsing strategies. Overall, labeled accuracy on a standard test set (from the CoNLL 2009 shared task), ignoring gold standard part-of-speech tags, increased significantly from 87.64% (baseline parser without hash kernel) to 89.40%.[10] We take this to indicate that a targeted and informed approach like the one we tested can have surprising effects even for a language that has received relatively intense consideration in the parsing literature.

Acknowledgements

We would like to thank Sandra Kübler, Yannick Versley and Yi Zhang for their support. This work was partially supported by research grants from the Deutsche Forschungsgemeinschaft as part of SFB 632 ”Information Structure” at the University of Potsdam and SFB 732 ”Incremental Specification in Context” at the University of Stuttgart.

References

Attardi, G. 2006. Experiments with a Multilanguage Non-Projective Dependency Parser. In *Proceedings of CoNLL*, pages 166–170.

Blum, A. 2006. Random Projection, Margins, Kernels, and Feature-Selection. In *LNCS*, pages 52–68. Springer.

Bohnet, B. 2009. Efficient Parsing of Syntactic and Semantic Dependency Structures. In *Proceedings of CoNLL 2009)*.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol.

Buchholz, Sabine and Erwin Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *In Proc. of CoNLL*, pages 149–164.

Carreras, X. 2007. Experiments with a Higher-order Projective Dependency Parser. In *EMNLP/CoNLL*.

[10] $\alpha = 0.01$, measured with a tool by Dan Bikel from www.cis.upenn.edu/˜dbikel/download/compare.pl

Crammer, K., O. Dekel, S. Shalev-Shwartz, and Y. Singer. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585.

Duchier, Denys and Ralph Debusmann. 2001. Topological dependency trees: a constraint-based account of linear precedence. In *Proceedings of ACL 2001*, pages 180–187, Morristown, NJ, USA. Association for Computational Linguistics.

Eisner, J. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of Coling 1996*, pages 340–345, Copenhaen.

Eisner, J., 2000. *Bilexical Grammars and their Cubic-time Parsing Algorithms*, pages 29–62. Kluwer Academic Publishers.

Hajič, J., M. Ciaramita, R. Johansson, D. Kawahara, M. Antònia Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, P. Straňák, M. Surdeanu, N. Xue, and Y. Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the 13th CoNLL-2009, June 4-5*, Boulder, Colorado, USA.

Hall, Johan and Joakim Nivre. 2008. A dependency-driven parser for German dependency and constituency representations. In *Proceedings of the Workshop on Parsing German*, pages 47–54, Columbus, Ohio, June. Association for Computational Linguistics.

Hinrichs, Erhard, Sandra Kübler, Karin Naumann, Heike Telljohann, and Julia Trushkina. 2004. Recent developments in linguistic annotations of the tüba-d/z treebank. In *Proceedings of the Third Workshop on Treebanks and Linguistic Theories*, pages 51–62, Tübingen, Germany.

Johansson, R. and P. Nugues. 2008. Dependency-based Syntactic–Semantic Analysis with PropBank and NomBank. In *Proceedings of the Shared Task Session of CoNLL-2008*, Manchester, UK.

Jurish, Bryan. 2003. A hybrid approach to part-of-speech tagging. Technical report, Berlin-Brandenburgische Akademie der Wissenschaften.

Kazama, Jun'Ichi and Jun'Ichi Tsujii. 2005. Maximum entropy models with inequality constraints: A case study on text categorization. *Machine Learning*, 60(1):159–194.

Kübler, Sandra. 2008. The PaGe 2008 shared task on parsing german. In *Proceedings of the Workshop on Parsing German*, pages 55–63, Columbus, Ohio, June. Association for Computational Linguistics.

Magerman, David M. 1995. Statistical decision-tree models for parsing. In *Proceedings of ACL 1995*, pages 276–283, Morristown, NJ, USA. Association for Computational Linguistics Morristown, NJ, USA.

McDonald, R. and F. Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *In Proc. of EACL*, pages 81–88.

Menzel, Wolfgang and Ingo Schröder. 1998. Decision procedures for dependency parsing using graded constraints. In *Proceedings of the COLING-ACL '98 Workshop on Processing of Dependency-Based Grammars*, pages 78–87.

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

Nivre, Joakim and Johan Hall. 2008. A dependency-driven parser for German dependency and constituency representations. In *Proceedings of the ACL Workshop on Parsing German*.

Nivre, J. and R. McDonald. 2008. Integrating Graph-Based and Transition-Based Dependency Parsers. In *ACL-08*, pages 950–958, Columbus, Ohio.

Nivre, Joakim, Jens Nilsson, Johan Hall, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with Support Vector Machines. In *Proceedings of CoNLL 2006*.

Nivre, J. 2003. An Efficient Algorithm for Projective Dependency Parsing. In *8th International Workshop on Parsing Technologies*, pages 149–160, Nancy, France.

Nivre, J. 2009. Non-Projective Dependency Parsing in Expected Linear Time. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 351–359, Suntec, Singapore.

Rahimi, A. and B. Recht. 2008. Random Features for Large-Scale Kernel Machines. In Platt, J.C., D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. MIT Press, Cambridge, MA.

Ratnaparkhi, Adwait. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP 1996*, volume 1, pages 133–142.

Schiller, Anne, Simone Teufel, and Christine Stöckert. 1999. Guidelines für das Tagging deutscher Textcorpora mit STTS (Kleines und großes Tagset). Technical Report August, Universität Stuttgart.

Schmid, Helmut. 1995. Improvements in part-of-speech tagging with an application to German. In *Proceedings of the ACL SIGDAT-Workshop*, volume 11.

Seeker, Wolfgang, Ines Rehbein, Jonas Kuhn, and Josef Van Genabith. 2010. Hard Constraints for Grammatical Function Labelling. In *Proceedings of ACL 2010*, Uppsala.

Titov, I. and J. Henderson. 2007. A Latent Variable Model for Generative Dependency Parsing. In *Proceedings of IWPT*, pages 144–155.

Yamada, H. and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of IWPT*, pages 195–206.