Stacking of Dependency and Phrase Structure Parsers

Richárd Farkas and Bernd Bohnet
Institute for Natural Language Processing
University of Stuttgart
{farkas,bohnetbd}@ims.uni-stuttgart.de

ABSTRACT
We investigate the stacking of dependency and phrase structure parsers, i.e. we define features from the output of a phrase structure parser for a dependency parser and vice versa. Our features are based on the original form of the external parses and we also compare this approach to converting phrase structures to dependencies then applying standard stacking on the converted output. The proposed method provides high accuracy gains for both phrase structure and dependency parsing. With the features derived from the phrase structures, we achieved a gain of 0.89 percentage points over a state-of-the-art parser and reach 93.95 UAS, which is the highest reported accuracy score on dependency parsing of the Penn Treebank. The phrase structure parser obtains 91.72 F-score with the features derived from the dependency trees, and this is also competitive with the best reported PARSEVAL scores for the Penn Treebank.

KEYWORDS: parsing, stacking, dependency parsing, phrase structure parsing.

1 Introduction

Both phrase structure and dependency parsers have developed considerably in the last decade, cf. (Nivre et al., 2004; McDonald et al., 2005b; Charniak and Johnson, 2005; Huang, 2008). The development has taken rather different directions as phrase structure parsers and dependency parsers employ different techniques to parse sentences. Phrase structure parsers usually apply probabilistic context free grammars and focus on the relationships among phrases. Dependency parsers use edge factored models for parsing that primarily model the interaction between the a head word and a dependent word. Second order graph-based parsers consider in addition for the decision on a dependency edge the interaction with siblings and grandchildren. Phrase structure and dependency parsers have both shown to be efficient and to provide accurate parsing results. Different parsing approaches have different strengths on distinct linguistic constructions, cf. (Nivre and McDonald, 2008).

In this paper, we exploit the difference of the representations of dependency and phrase structure parses and the divergence in the parsing techniques. We use features derived from the 1-best automatic phrase structure parse to augment the feature set of the dependency parser and vice versa. This way of combining parsers proved to be effective and provides a substantial accuracy gain for both parsing approaches.

Our ultimate objective is to advance one parser by a second one. This motivation is different from previous approaches for combining phrase-structure and dependency parsers (cf. (Klein and Manning, 2003; Carreras et al., 2008; Rush et al., 2010)) which aimed to achieve a joint optimum of the two approaches. Our proposal has three advantages over previous work. (i) Our approach is simple to implement by defining new feature templates, yet very effective. (ii) It does not require a joint representation of different parse structures. (iii) The participating dependency and phrase-structure parsers can be easily replaced, and each can be developed and optimized independently.

In our experiments both parsers utilize the same (training) data set – having two different representations – without having access to external information. Thus, our results indicate that state-of-the-art parsers can be still further improved without additional data and that there may be approaches that reach even higher accuracy levels in a single pass. At the dependency parser framework, we also investigated whether our approach retains its added value over other extensions and we answer the research questions: Is there a significant improvement still possible with two stacked dependency parsers? What is the impact on top of the results which utilize external information?

We also describe the findings of a manual contribution analysis of the second parser. This discussion identifies linguistic constructions which are responsible for the improvements and it brings up ideas how the parser approaches themselves can be improved.

The contributions of this paper are as follows:

- We propose the stacking of dependency and phrase-structure parsers.
- We report outstanding scores on the Penn Treebank and on the German Tiger Treebank for stacking a phrase structure parser on and a dependency parser and vice versa. The added value of the approach in the dependency parsing environment is still considerable in the presence of other extensions.
- We give explanations why the proposed stacking approach works.

2 Related Work

A number of studies have addressed feature-rich dependency and phrase structure parsing, cf. (Nivre et al., 2004; McDonald et al., 2005b; Charniak and Johnson, 2005; Huang, 2008).

The two main approaches to **dependency parsing** are transition-based dependency parsing (Yamada and Matsumoto, 2003; Nivre, 2003; Titov and Henderson, 2007), and graph-based dependency parsing (Eisner, 1996; McDonald et al., 2005a; Carreras, 2007; Rush et al., 2010).

The most successful supervised **phrase structure parsers** are feature-rich discriminative parsers which heavily depend on an underlying PCFG (Charniak and Johnson, 2005; Huang, 2008). These approaches consist of two stages. At the first stage they apply a PCFG to extract possible parses, then at the second stage select the best parse from the set of possible parses (i.e. rerank this set) employing a large feature set (Collins, 2000; Charniak and Johnson, 2005).

There is little related work on **combining the two approaches**. Some generative parsing approaches have exploited the differences between phrase structure and dependency parsers. For instance, Klein and Manning (2003) introduced an approach where the objective function is the product of the probabilities of a generative phrase structure and a dependency parser. Model 1 of Collins (2003) is based on the dependencies between pairs of head words. On the other hand, the related work on this topic for discriminative parsing is sparse, and we are only aware of the following works. Carreras et al. (2008) and Rush et al. (2010) introduced frameworks for joint learning of phrase and dependency structures, and showed improvements on both tasks for English. These frameworks require special formulation of – one or both – parsing approaches while our approach allows the usage of arbitrary dependency parsers and any feature-based phrase structure parser.

Our motivation differs from these solutions as we focus on advancing one approach rather than achieving a joint optimum. Our approach can be regarded as a special stacking procedure (Nivre and McDonald, 2008), specifically, the stacking of a phrase structure parser with a dependency parser. In our previous work (Farkas et al., 2011), we reported results with a stacking approach for phrase structure parsing evaluated on a German corpus. We focus herein on the reverse direction as well, i.e. we define features for dependency parsers, we report performance outstanding scores on English for both directions, compare to the state-of-the-art results and carried out a detailed analysis of the stacking's contributions.

Wang and Zong (2010) introduced a procedure that exploits dependency parses to improve a phrase structure parser. They used automatic dependency parses for pruning the chart of a phrase structure parser and reported a significant improvement. One of our feature templates for the phrase structure parser can be regarded as a generalization of this approach.

3 Dependency Parser Exploiting Phrase Structure Parses

In this study we focus on graph-based parsers, however, our features from the phrase structures can also be adapted to a transition-based parser. **Graph-based dependency parsers** decompose the dependency structure into *factors*. Each factor of the first order graph-based parser corresponds to a dependency edge. McDonald et al. (2005a) first used second order factors which incorporates siblings of the head and dependent.

The second order algorithm of Carreras (2007) uses three second order factors: the sibling, the leftmost and rightmost grandchildren. The edge labeling is an integral part of the algorithm, which requires an additional loop over the labels. This algorithm has a complexity of $O(n^4L)$.

Koo and Collins (2010) presented a parser that can evaluate factors of three edges.

In this paper, we utilize the state-of-the-art parser of Bohnet (2010), a graph-based parser which employs online training with a perceptron which employs the MIRA update (Crammer and Singer, 2003). The parser contains a feature function for the first order factor, one for the sibling factor, and one for the grandchildren. We integrated in each of the functions features representing information on the phrase structure.

3.1 Features Defined on Phrase Structure Parses

We explore new feature templates to include features from the phrase structures. These templates are defined on the phrase structures themselves which is an important difference compared to previous work on joint parsing. Joint parsing requires a joint representation of phrase and dependency structures and is traditionally carried out by – explicitly or implicitly – converting the phrase structures into bilexical dependencies. For a comparative analysis of these two approaches please refer Section 6.3. Here we introduce our feature template utilizing the original phrase structures.

The feature templates provide information for the dependency parser about the embedding of the nodes in the phrase structure. A first order factor consists of the head and the dependent. A second order factor comprises of two connected edges, which additionally consist of a sibling or grandchild. The feature templates represent the location of the nodes in the phrase structure and the label of the phrase involved. We use the following abbreviations to define the feature templates: `L` represents the edge label; `hP`, `dP`, and `xP` the part-of-speech tag of the head, that of the dependent and the sibling or grandchild. The `hPoPS`,`dPoPS`, `xPoPS` denote the part-of-speech tag provided by the phrase structure parser for the head, dependent, and sibling or grandchild, respectively.

`hPS`, `dPS`, `xPS` denote the label of the phrase where the corresponding terminal node is embedded, respectively. `hPPS`, `dPPS`, `xPPS` denote the phrase that contains the phrase of the corresponding terminal of the head, dependent and sibling/grandchildren, respectively. The **r**elative **l**ocation, `rl`, of the head and dependent in the phrase structure: `rl=Phd`, if the head and dependent are terminals of the same phrase; `rl=Ph->Pd`, if the dependent is in a subphrase of the phrase of the head; `rl=Pd->Ph` denotes the inverse relation; otherwise `rl=Pd?Ph`. `rlx` stands for the relative location of the head and dependent, grandchild or sibling in the sentence, e.g. if the head is before or after the dependent in the sentence `rlx=h<d` or `rlx=d>h`. We count all pairwise combinations of the above nodes.

1. first order features:
   `L+hP+dP+hPS+rl`, `L+hP+dP+dPS+rl`, `L+hP+dP+hPS+dPS+rl`,
   `L+dP+hPS+hPPS+dPS+rl`, `L+dP+hP+dPPS+hPS+rl`, `L+hPoPS+dPoPS`.

2. sibling features:
   `L+hP+dP+xP+hPS+rlx`, `L+hP+dP+xP+dPS+rlx`, `L+hP+dP+xP+xPS+rlx`,
   `L+hPS+dPS+xPS+rlx`, `L+xP+hPS+dPS+xPS+rlx`, `L+xP+hPS+dPS+xPS+xPPS+rlx`,
   `L+xP+hPPS+dPS+xPS+rlx`, `L+xP+hPPS+dPS+xPS+xPS+rlx`,
   `L+hPoPS+dPoPS+xPoPS`.

3. We use as grandchildren features the same feature set as for the sibling features and replace x by the grandchild.

4 Phrase Structure Parser Exploiting Dependency Parses

Phrase structure parsers use a PCFG to extract possible parses. The full space of possible parses cannot be iterated through in practice, and they are usually pruned as a consequence. The *n-best list parsers* keep just the 50-100 best parses according to the PCFG (Charniak and Johnson, 2005). Other methods remove nodes and hyperedges whose posterior probability is under a pre-defined threshold from the forest (chart) (Huang, 2008). The task of the second stage is to select the best parse from the set of possible parses (i.e. rerank this set). These methods employ a large feature set (usually a few millions features) (Collins, 2000; Charniak and Johnson, 2005). The n-best list approaches can straightforwardly employ local and non-local features as well because they decide at the sentence-level (Charniak and Johnson, 2005) while involving non-local features is more complicated in the forest-based approaches and studies show only a minor empirical advantage of the latter approach (Huang, 2008; Wang and Zong, 2011). In this study, we experiment with n-best list reranking using a Maximum Entropy machine learning model and our goal is to investigate the extension of the standard feature set of these models by features extracted from the automatic dependency parse of the sentence in question.

4.1 Features Defined on Dependency Parses

The objective of the features defined for phrase structure parsing is to characterize the divergence/similarity of constituents with the corresponding part of the automatic (1-best) dependency parse of the sentence in question. We defined three feature templates for representing hyperedges (i.e. CFG rules applied over a certain span of words). We illustrate them on two possible subparses of Figure 1.

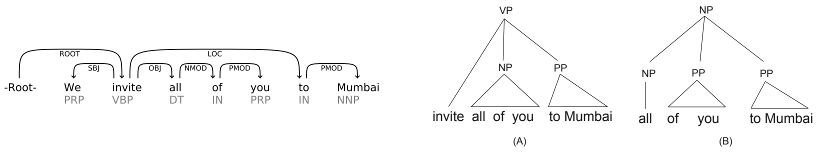

Figure 1: A dependency parse and two hyperedges (*A* and *B*) of the phrase structure chart for the introduction of feature templates.

outArc features are counting the dependency arcs which "go out" from the constituent in question. More precisely we count the words within the span whose parent in the dependency tree lays outside the span of words in question. We use the absolute count and the ratio of outArcs among the words of the span. The more arcs go out, the further away is the dependency subtree over the words of the constituent from a dominating subtree. Hence, these features try to capture the "phraseness" of the span of words in question based on the dependency tree. The example *A* in Figure 1 gets `outArc=1` and `outArcRatio=1/6` as only the parent of *invite* lays outside the constituent. For *B* we have `outArc=2` and `outArcRatio=2/5` because of *all* and *to*.

POSRel features indicate the dependencies which occur in the dependency parse (explicitly) as well as in the phrase-structure parse (implicitly). For this we gather the daughter constituents whose lexical head is linked in the (undirected) dependency tree to the head of the parent constituent. We define features from them using the pair of the two heads' POS tag and a triplet using the POS tags and the corresponding dependency label. For *A* we have the following binary valued features: `VBP-DT`, `VBP-DT-OBJ` (from *invite–all*), `VBP-IN`, `VBP-IN-LOC` (from *invite–to*) as both daughter attachments have the corresponding arcs in the dependency tree. For *B* we do not extract features for the second PP daughter attachment as the lexical head of the parent (*all*) and the lexical head of the daughter (*to*) are not linked in the dependency parse.

ConstRel features are similar to `POSRel` but use the constituent labels rather than the POS tags of the heads. Thus, for *A* we have `VP-NP`, `VP-NP-OBJ` (from *invite–all*), `VP-PP`, `VP-PP-LOC` (from *invite–to*).

We also investigated the role of case and grammatical functions of the phrase structure parser in German and extended the `POSRel` and `ConstRel` feature sets by adding this information to the labels.

Note that the value of `outArc` is 1 iff the word span in question has a dominating dependency subtree in the automatic parse. Wang and Zong (2010) prune hyperedges with `outArc`$\neq 1$ thus this feature can be regarded as a generalization of their approach.

These features are similar in nature to the lexical head-based features of Charniak and Johnson (2005) and Versley and Rehbein (2009) but their role is different. Those dependency-like features try to describe the local hyperedges using the internal state of the parsing approach while we exploit here a globally (i.e. sentence-level) optimized dependency parse tree which represents external knowledge to the parser.

5 Experimental Setting

5.1 Data Sets

We used the Penn Treebank (section 23 served as test set and section 22 and 24 were used as development set in the phrase-structure and dependency experiments, respectively) with the Penn2Malt converter, and the head-finding rules of Yamada and Matsumoto (2003) for our experiments on English because most of the published state-of-the-art results were obtained on these datasets and conversion. For German, we used the Tiger Treebank and followed the split of the Parsing German (PaGe) shared task (Kübler, 2008) for phrase structure parsing. For dependency parsing, we used the dependency corpora of the CoNLL-2009 shared task, cf. (Hajič et al., 2009) (which consists of automatically converted trees from a part of the Tiger Treebank).

The phrase structure parsers traditionally conduct POS tagging during parsing (the different POS alternatives are on the chart). For obtaining POS tags for the dependency parsing experiments, we jackknifed the training data 10 fold, i.e. we trained the POS tagger on nine folds and tagged the tenth fold. On the English training set, the POS tagger of Toutanova et al. (2003) achieves an accuracy of 97.1, on the development set, 97.2 and the test set, 97.3. For German, we used a SVM-based tagger and we obtained on the training set an accuracy of 97.2, on the development set, 97.5 and on the test set, 97.4.

We followed the same jackknifing procedure in the stacking experiments in order to obtain the parses from which the features are extracted. We generated the dependency and constituent parses of the training corpus by training the respective parser on nine folds and parsing the tenth fold. During parsing the test set and development set we utilized the parses from the second parser which was trained on its entire training corpus.

5.2 Implementation Details

For our experiments, we utilize the second-order graph-based parser of the **dependency parsers** of Bohnet (2011)[1]. The second-order parser employs online training with a perceptron (Crammer and Singer, 2003) and it is based on a Hash Kernel. We employed this parser since it is quick to train, provides useful labeled dependency trees and achieves state-of-the-art accuracies. We used the transition-based parser that is included in the same package for our experiments as well.

For the **phrase structure parsing** experiments, we also employed state-of-the-art parsers. We used the first-stage parser of Charniak and Johnson (2005) for English and Bitpar (Schmid, 2004) for German. The latter employs a grammar engineered for German (Farkas et al., 2011). At the second stage, we used the 50 and 100-best parses to rerank and filtered out rare features (which occurred in less than 5 sentences). We employed the ranking MaxEnt implementation of the MALLET package (McCallum, 2002) and optimized the L2 regularizer coefficient on the development set.

5.3 Baseline Feature Sets

For conducting baseline **dependency parsing** experiments, we employed the feature set of Bohnet (2010). For the stacking of graph-based and transition-based dependency parsers, we used the feature template definitions from Nivre and McDonald (2008) which was extended by grandchildren factors (similarly to Torres Martins et al. (2008)).

For **phrase structure reranking** we utilized the features of Charniak and Johnson (2005) and for German we reimplemented the feature templates of Versley and Rehbein (2009) as baseline feature sets. The latter is the state-of-the-art feature set for German, which consists of features constructed from the lexicalized parse tree and its typed dependencies along with features based on external statistical information (like the clustering of unknown words according to their context of occurrences and PP attachment statistics gathered from the automatic POS tagged DE-WaC corpus, a 1.7G words sample of the German-language WWW).

5.4 Evaluation Metrics

For the evaluation of the dependency parses, we use the standard labeled (LAS) and unlabeled attachment scores (UAS) excluding punctuation for English and including punctuation for German. We use the predicted POS tags everywhere thereafter.

We use the standard `evalb` implementation of PARSEVAL on every sentence without length limitation as evaluation metric for phrase structure parsers. As the grammatical functions of constituents are important for downstream applications – especially in German –, we also report PARSEVAL scores on the conflation of constituent labels and grammatical functions (the scores in brackets in Table 3). We have to mention that our F-values are not directly comparable to the official results of PaGe Shared Task – which was our original goal – because the evaluation metric had a special implementation for calculating the F-measure (which differs from `evalb` for example in handling punctuation marks) and it used gold-standard POS tags in the input (which we thought to be unrealistic).

[1]Downloadable from http://code.google.com/p/mate-tools/

6 Results

6.1 Oracle Experiments

In order to validate the value of our new feature templates, we carried out the following oracle experiment. We used the gold standard parse trees from the second parser for training and parsing, i.e. we extracted features from the gold standard phrase structure parses for dependency parsing and vice versa. Our dependency parser achieved an UAS of 98.52 and LAS of 98.17 on the English test set. The re-ranking phrase structure parser achieved 95.78 where the oracle score – i.e. selecting the best candidate from the candidate list – is 96.83. Hence using the gold standard trees of the second parser yield scores close to the theoretical upper bound, which empirically proves that our new feature templates can effectively capture the information present in the other syntactic representation.

6.2 Dependency Parsing Results

Table 1 shows the results achieved on the test and development sets of English and German.

Our baseline parsers here are the transition-based parser (baseline T) and graph-based parser (baseline G) of (Bohnet, 2011). The row "G+T features" presents stacking results for the transition-based and graph-based parser. In this setting, the graph-based parser uses the output of the transition-based parser. This setting shows already a high accuracy improvement of 0.44 and obtained an unlabeled accuracy score of 93.5 in English. This result shows that the stacking idea of Nivre and McDonald (2008) works well on this higher accuracy level obtained with the offspring of their parsers.

The row "G+P features" shows the results when the graph-based parser exploits the features extracted from the phrase structure parser. The unlabeled accuracy score achieved the current best results for this data set with 93.95 UAS. When we use the output of the transition-based parser and the phrase structure parser as input to the graph-based parser (G+P+T features), we are able to obtain an even higher accuracy score of 94.04. Exploiting phrase structure parses achieves a gain in UAS of 0.89 and 0.54 employing them as an extension of the graph-based and the stacked dependency parser G+T. The added value of the features from the transition-based dependency parses on top of the extended feature set (i.e. the difference between G+P and G+P+T) is minor – and there is a decrease at 2 out of the 8 settings. These results indicate that the features gathered from the phrase structure parse contain almost all of the information which can be gathered from the transition-based dependency parse.

The two last rows show scores when we use in addition cluster-based features similarly to Koo et al. (2008). The cluster-based feature provided an improvement of 0.17 on top of the results of the graph-based parser (G+C features). The last row (G+P+T+C features) shows the scores for the features from all sources where we finally obtain 94.14 UAS.

Similarly to the results for English, we can also report large improvements for the German Tiger Treebank. With the G+T features, we gain 0.24 UAS and 0.71 with the "G+P features" compared with the "baseline G". When we utilized each feature set, the parser achieved 91.88 UAS and 89.90 LAS (0.81 improvements in UAS), which is the highest reported accuracy score for this data set.

	English		German	
	Devel.	Test	Devel.	Test
baseline T	91.46/90.10	92.71/91.56	90.27/87.76	90.39/88.05
baseline G	92.09/90.81	93.06/91.95	90.37/88.01	91.02/88.81
G+T features	92.34/90.96	93.50/92.53	90.45/88.29	91.26/89.13
G+P features	92.81/91.71	93.95/92.99	90.64/88.47	91.73/89.50
G+P+T features	92.83/91.70	94.04/93.10	90.70/88.54	91.69/89.69
G+C features	92.14/90.87	93.23/92.15	90.53/88.31	91.40/89.19
G+P+T+C features	92.96/91.80	94.14/93.17	90.73/88.50	91.88/89.90

Table 1: Dependency parsing results achieved by the enriched feature sets. The accuracy scores are statistically significant between the baseline G and G+T, G+P, and G+P+T+C with $p < 0.001$ and between G+P and G+P+T+C with $p<0.05$. We used two sample t-test.

6.3 Stacking of Converted Parses

An alternative approach for stacking phrase structures into a dependency parser is to convert the phrase structures into dependency trees and extract features from this representation. We investigated this approach on the English dataset[2]. We used the Penn2Malt converter on the automatic phrase structure parses and then employed the same dependency stacking features which were used for "G+T features" and were described in Section 5.3. Table 2 repeats the scores from Table 1 achieved by the baseline parser and by the feature templates defined directly on phrase structures and compares these scores with the results achieved by the conversion-based stacking.

	English	
	Devel.	Test
baseline G	92.09/90.81	93.06/91.95
G+P converted	92.48/91.23	93.74/92.73
G+P direct	92.81/91.71	93.95/92.99

Table 2: Results achieved by utilizing features defined directly on phrase structures vs. defined on converted trees. The accuracy scores are statistically significant between the two approaches with $p<0.05$. We used two sample t-test.

6.4 Phrase Structure Parsing Results

Table 3 summarize the results achieved by phrase structure parsers[3]. In the reranking approach we used 50-best lists for English and 100-best lists for German as it is standard in previous work.

The "generative" row stands for the results achieved by the first-stage parsers. The "oracle" score is the highest available score in the 50-best and 100-best lists, thus serve as an upper

[2]Unfortunately, we do not have access to the conversion scripts for German. As the conversion in the reverse direction (i.e. converting dependency parses into phrase structures) is not straightforward we restricted ourself to phrase structure to dependency conversion and the English dataset.

[3]Note that the difference in German scores between this table and the scores reported in our previous study (Farkas et al., 2011) is because we report score on each sentences – and compare our results here against re-trained baseline parsers – while we limited the evaluation to sentences shorter than 40 tokens in the previous study – in order to be able to compare those results against previously published scores on the same corpus.

bound for rerankers. The last three rows contain the results of reranking on top of first-stage parsers employing the baseline[4], only the new features defined from dependency parses (dep) and their union (bl+dep).

	English		German	
	Devel.	Test	Devel.	Test
generative	88.86	89.71	77.83 (66.88)	76.81 (65.76)
oracle	95.90	96.83	86.18 (76.10)	83.37 (73.07)
baseline features	90.59	91.38	79.76 (67.99)	78.41 (67.05)
dep features	89.90	90.66	79.82 (69.28)	78.01 (67.46)
bl+dep features	90.92	91.72	80.90 (70.39)	79.45 (68.73)

Table 3: Phrase structure parsing PARSEVAL scores achieved by utilizing various feature sets. The figures in brackets for German are PARSEVAL scores including grammatical functions. The accuracy scores are statistically significant between "baseline features" and "bl+dep features" with $p<0.05$. We used two sample t-test.

As Table 3 shows, the reranking parser using only our features constructed from the automatic dependency parse of the sentence achieved a 1.0 improvement on the English development and test sets, respectively and they added a value of 0.3 over the baseline feature set as well.

The results for German are even more convincing. The simple dependency parse-based features are competitive with the German-specific feature set of Versley and Rehbein (2009). Moreover these two feature sets have a certain level of diversity as their union could achieve significantly better results (over 1.0 point of improvements), than any of them alone. The added value of the dependency parse-based features is higher if we consider the evaluation metric which also takes into account the grammatical function labels as well.

6.5 Comparison to previous work

Table 4 shows labeled and unlabeled accuracy scores of previous work reported for the Penn2Malt conversion with the head finding rules of Yamada and Matsumoto (2003). Carreras et al. (2008) (CCK) combine a dependency parsing decoder with parsing based on TAG. Koo and Collins (2010) (Koo'10) used a parser with third order factors and obtained an unlabeled accuracy of 93.04. Martins et al. (2010) introduced a parser (Turbo) with even higher accuracy (93.26 UAS), which is one of the highest accuracy scores for a parser that does not employ additional sources of information.

The following approaches use additional sources of information. Suzuki et al. (2009) (SICC) use a second order dependency parser that employs siblings and grandchildren factors. They apply the semi-supervised learning approach of Suzuki and Isozaki (2008) to dependency parsing and include additionally the cluster-based features of Koo et al. (2008) (KCC).

The improvement of the accuracy achieved by our stacking approach is surprisingly high. We achieved an error reduction of 16.5% (an absolute gain of 1.1 percentage points) over our baseline and it outperforms the best reported scores with combining dependency and phrase structure parses (Carreras et al., 2008) by 0.6.

[4]The small difference between the original Brown parser and reranking with baseline features is due to the different MaxEnt implementations employed.

	English (UAS)			German (LAS)	
	Devel.	Test		Devel.	Test
CCK'08	92.5	93.5	GHPT'09		87.28
Koo'10		93.04	B'10		88.06
Turbo'10		93.26			
This work (G+P+T)	92.14	94.04		88.54	89.69
KCC'08	91.85	93.16			
SICC'09		93.8			
This work (G+P+T+C)	92.96	94.14		88.50	89.90

Table 4: Comparison to previous work. We report UAS only for the English Penn Treebank since usually unlabeled scores were published and labeled attachments scores only for the German Tiger Treebank since in the CoNLL shared task 2009 only labeled attachment scores were reported.

For German the two top scoring parsers of the CoNLL 2009 shared task on Syntactic and Semantic Dependency parsing were the parser of Gesmundo et al. (2009), who had the best overall scores and the parser of Bohnet (2010), which performed best on English and German. However we note that directly comparing these results to our ones is not fully fair since the POS tagging accuracy of the data sets were lower than 95.5 at the CoNLL 2009 shared task.

Table 5 sketches the **phrase-structure parsing** results in the context of previously published results. We retrained the current version of the Charniak and Johnson (2005) parser[5] (C&J) for English. We also cite the scores reported by Rush et al. (2010) who also exploited phrase structure and dependency parsers together. Our final scores are competitive with the best reported supervised PARSEVAL score in the Penn Treebank by (Huang, 2008) who also extended the feature set of Charniak and Johnson (2005) and applied forest-based reranking.

	English			German	
	Devel.	Test		Devel.	Test
C&J	90.60	91.36	Berkeley	76.60 (65.25)	76.05 (64.00)
Rush'10	–	90.7	PyNLP	76.76 (66.29)	76.27 (65.21)
Huang'08	–	91.69			
This work (bl+dep)	90.92	91.72		80.90 (70.39)	79.45 (68.73)

Table 5: Comparison to previous work. Our method significantly outperforms state-of-the-art systems.

For German, we trained the current version of the Berkeley parser[6] (Petrov et al., 2006) and PyNLP[7] parsers (Versley and Rehbein, 2009). Note that the results of the generative Berkeley parser is competitive with the PyNLP parser which utilizes reranking, while the first-stage parser, which we employ here – using a fine-tuned German-specific tree annotation schema – outperforms both of them. The discriminative approach with the baseline feature set we employed gave an improvement of 2 percentage points over the first-stage parser and the features gathered from dependency parsers could add an additional 1 point to this.

[5]http://www.cog.brown.edu/ mj/Software.htm
[6]http://code.google.com/p/berkeleyparser/
[7]https://bitbucket.org/yannick/pytree

7 Discussion

We manually analyzed the outputs of the parsers in order to identify linguistic constructions where one or another parser performed better. Our findings help understand the reasons why the approach works and they might suggest ways of improving data-driven parsers as well.

7.1 Why did the phrase structures help dependency parsing?

In order to find the explanation why phrase structure parse-based features help the dependency parser we manually investigated the sentences of the English development set where the parsers with and without these features gave different output.

We found that the most frequent error fixing categories are related to longer phrases, typically noun phrases and named entities. For example, the phrase structure parser could recognize the phrase boundaries of *US Today 's total paid ad pages* and *St. Therese , Quebec* and with the phrase structure-based features the dependency parser was able to fix its errors here. Note that in the latter example if the parser analyses the entity as two separated entities *St. Therese* and *Quebec*, it introduces 3 attachment errors (for *Therese*, for the comma and for *Quebec*). Besides noun phrases the parses of institutionalized phrases like *with or without* and *rather than to* were also considerable improved. These improvements can be probably attributed to the phrase-based thinking of the second parser.

Phrase structure-based features improved considerably the accuracy of coordinations. It is a known issue that the representation of coordination in the dependency structure has an impact on the accuracy. The Penn2Malt converter builds dependency structures where the conjunction is the head. Nilsson et al. (2007) showed that this representation of coordinations yields a lower parsing accuracy – across four languages – than a representation where the conjunction and the members of the coordination form one chain. Our features defined on the phrase structure trees can be regarded as a third type of representation of coordination. Although these features help in the current setting we expect a smaller contribution if a different dependency representation (like the chain type) is targeted.

The counting of the errors – or error fixes – related to coordination is not straightforward because besides the wrong attachment of a `CC` it introduces several attachment errors in the construction. For instance, we observed several article attachment errors related to this as well. For example in *the agreement and consent decree* the article can be attached to the *agreement* or to the *decree* according to the analysis of the coordination.

The rest of the improvements is heterogenous but we observed two other quite frequent error types. The first one is the attachment to a verb if multiple verbs are present in the sentence, most typically the choice between the auxiliaries and the main verbs. For instance in *having just passed* the dependency parser attached *just* to the main verb while was able to correctly attach it to the auxiliary with the help of phrase structure-based features. We counted down the attachment errors of the dependency parser where it chose a wrong verb. The phrase structure-based features could eliminate 245 such errors while introduced 117 new attachment errors.

A related interesting issue is that the incorporation of the phrase structure-based features into the graph-based parser improves the selection of the sentence's root by almost 1 percentage point (the F-score of the `ROOT` label increased from 96.24 to 97.1). This might be an artifact of the two-stage approach of the phrase structure parser, i.e. the reranking second stage could choose from candidates having different roots. We intend to more deeply investigate this issue in the future as it seems to be promising to design a reranking phase for dependency parsers which can select from parses with different roots.

7.2 Why did the dependency structures help phrase structure parsing?

We followed the methodology of contribution analysis for phrase structure parses and we manually investigated the sentences of the English development set where the discriminative parsers with and without these features gave different output. We identified two categories of improvement. The first one is the "attachment" of adverbs and adjectives, especially in the case of collocational modifiers like *more like* and *many more*. The second phenomenon where the dependency parse-based features could eliminate a considerable amount of errors – much more than was introduced – is PP attachment. The explanation for both of these contributions might be the "lossless" lexicalization of the dependency parsers. Although the PCFG parser of Charniak (2000) is lexicalized[8] and the re-ranking stage is deeply lexicalized, using only 50 candidate parses and a frequency-based feature pruning implies some extent of information loss while dependency parsers can exploit very rare lexical cues.

7.3 What is the difference between stacking directly and stacking after conversion?

We also manually compared the parses of the two stacking approaches, i.e. the one where the features are defined directly on phrase structures versus the other where the features are defined on a dependency tree which were automatically converted from the phrase structures (see Section 6.3). Although the difference between the two approaches is significantly different we cannot conclude that the first one is clearly better than the second one. There are 214 sentences in the development set where the direct appraoch achieves higher attachment scores than the other while there are also 197 sentences where its scores were worse.

Interestingly, the most frequent linguistic constructions which are responsible for these differences are related to conjunctions at both the directions. We found that in the case of noun phrase coordinations, the features defined directly on the phrase structures are more useful than after conversion. This further supports our earlier discussion that our features as a representation type of coordination is more useful than the "conjunct is the head" representation used in the Penn2Malt conversion.

On the other hand, in the case of compound sentences (i.e. the conjunction of independent clauses) the converted parses contributed frequently more than features directly defined on the phrase structures (this is expressed also by the F-score of the `ROOT` label which is 0.3 higher in the first case). The cause for this effect is that the phrase structure puts the two (or more) independent clauses to same level while the dependency tree – at least in the Penn2Malt conversion – has to label a single token as the root of the sentence. The root selection among the heads of the independent clauses follows a deterministic rule during the conversion process, thus the converted trees do not face the problem of root ambiguity in these compound sentences.

8 Conclusions

We proposed here the stacking of dependency and phrase structure parsers. We introduced a feature set for describing phrase structures for data-driven dependency parsers and dependency

[8]However, it utilizes just the head of the parent instead of bi- (or higher order) lexicalization.

parses for discriminative phrase structure parsers. We investigated the performance of our stacking approach on English and German and we reported a significant improvement over the state-of-the-art parsers in both directions. We believe that the key factor of this success is that we designed our framework to advance one parser exploiting the other, rather than optimizing a joint objective function.

In our experiments both parsers utilize the same (training) data set without having access to external information. We argue that our results – especially the gain at dependency parsers – indicate that state-of-the-art parsers can be still further improved without additional data. To facilitate these further developments, we manually analyzed several parses of different approaches and identified linguistic phenomena where the gain was considerable. For instance, for the dependency parsers these phenomena are named entities, longer noun phrases, coordination and root selection. Investigating them more deeply could probably reveal parsing techniques which can yield considerable added value to current procedures.

Acknowledgement

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) via the SFB 732 "Incremental Specification in Context", projects D4 (PI Helmut Schmid) and D8 (PI Jonas Kuhn).

References

Bohnet, B. (2010). Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING*, pages 89–97.

Bohnet, B. (2011). Comparing advanced graph-based and transition-based dependency parsers. In *International Conference on Dependency Linguistics*, pages 282–289.

Carreras, X. (2007). Experiments with a Higher-order Projective Dependency Parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 957–961.

Carreras, X., Collins, M., and Koo, T. (2008). Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 132–139.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 173–180.

Collins, M. (2000). Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 175–182.

Collins, M. (2003). Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29:589–637.

Crammer, K. and Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Eisner, J. M. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 340–345.

Farkas, R., Bohnet, B., and Schmid, H. (2011). Features for phrase-structure reranking from dependency parses. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 209–214.

Gesmundo, A., Henderson, J., Merlo, P., and Titov, I. (2009). A Latent Variable Model of Synchronous Syntactic-Semantic Parsing for Multiple Languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009)*.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., and Zhang, Y. (2009). The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the 2009 CoNLL Shared Task*, pages 1–18.

Huang, L. (2008). Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08*, pages 586–594.

Klein, D. and Manning, C. D. (2003). Fast exact inference with a factored model for natural language parsing. In *Proceedings of Advances in Neural Information Processing Systems*, volume 15.

Koo, T., Carreras, X., and Collins, M. (2008). Simple semi-supervised dependency parsing. In *Proceedings of ACL*, pages 595–603.

Koo, T. and Collins, M. (2010). Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11.

Kübler, S. (2008). The PaGe 2008 shared task on parsing german. In *Proceedings of the Workshop on Parsing German*, pages 55–63.

Martins, A. F. T., Smith, N. A., Xing, E. P., Aguiar, P. M. Q., and Figueiredo, M. A. T. (2010). Turbo parsers: dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 34–44.

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

McDonald, R., Crammer, K., and Pereira, F. (2005a). Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98.

McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. (2005b). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530.

Nilsson, J., Nivre, J., and Hall, J. (2007). Generalizing tree transformations for inductive dependency parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT*, pages 149–160.

Nivre, J., Hall, J., and Nilsson, J. (2004). Memory-based dependency parsing. In Ng, H. T. and Riloff, E., editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 49–56.

Nivre, J. and McDonald, R. (2008). Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440.

Rush, A. M., Sontag, D., Collins, M., and Jaakkola, T. (2010). On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

Schmid, H. (2004). Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings of Coling 2004*, pages 162–168.

Suzuki, J. and Isozaki, H. (2008). Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of ACL-08*, pages 665–673, Columbus, Ohio. Association for Computational Linguistics.

Suzuki, J., Isozaki, H., Carreras, X., and Collins, M. (2009). An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 551–560.

Titov, I. and Henderson, J. (2007). A latent variable model for generative dependency parsing. In *Proceedings of IWPT*, pages 144–155.

Torres Martins, A. F., Das, D., Smith, N. A., and Xing, E. P. (2008). Stacking dependency parsers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 157–166, Honolulu, Hawaii. Association for Computational Linguistics.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Conference on North American chapter of the Association for Computational Linguistics*, pages 252–259.

Versley, Y. and Rehbein, I. (2009). Scalable discriminative parsing for german. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 134–137.

Wang, Z. and Zong, C. (2010). Phrase structure parsing with dependency structure. In *Proceedings of COLING*, pages 1292–1300.

Wang, Z. and Zong, C. (2011). Parse reranking based on higher-order lexical dependencies. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1251–1259.

Yamada, H. and Matsumoto, Y. (2003). Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of International Conference on Parsing Technologies*, pages 195–206.