Graph Transformations in Data-Driven Dependency Parsing

Jens Nilsson
Växjö University
jni@msi.vxu.se

Joakim Nivre
Växjö University and
Uppsala University
nivre@msi.vxu.se

Johan Hall
Växjö University
jha@msi.vxu.se

Abstract

Transforming syntactic representations in order to improve parsing accuracy has been exploited successfully in statistical parsing systems using constituency-based representations. In this paper, we show that similar transformations can give substantial improvements also in data-driven dependency parsing. Experiments on the Prague Dependency Treebank show that systematic transformations of coordinate structures and verb groups result in a 10% error reduction for a deterministic data-driven dependency parser. Combining these transformations with previously proposed techniques for recovering non-projective dependencies leads to state-of-the-art accuracy for the given data set.

1 Introduction

It has become increasingly clear that the choice of suitable internal representations can be a very important factor in data-driven approaches to syntactic parsing, and that accuracy can often be improved by internal transformations of a given kind of representation. This is well illustrated by the Collins parser (Collins, 1997; Collins, 1999), scrutinized by Bikel (2004), where several transformations are applied in order to improve the analysis of noun phrases, coordination and punctuation. Other examples can be found in the work of Johnson (1998) and Klein and Manning (2003), which show that well-chosen transformations of syntactic representations can greatly improve the parsing accuracy obtained with probabilistic context-free grammars.

In this paper, we apply essentially the same techniques to data-driven dependency parsing, specifically targeting the analysis of *coordination* and *verb groups*, two very common constructions that pose special problems for dependency-based approaches. The basic idea is that we can facilitate learning by transforming the training data for the parser and that we can subsequently recover the original representations by applying an inverse transformation to the parser's output.

The data used in the experiments come from the Prague Dependency Treebank (PDT) (Hajič, 1998; Hajič et al., 2001), the largest available dependency treebank, annotated according to the theory of Functional Generative Description (FGD) (Sgall et al., 1986). The parser used is MaltParser (Nivre and Hall, 2005; Nivre et al., 2006), a freely available system that combines a deterministic parsing strategy with discriminative classifiers for predicting the next parser action.

The paper is structured as follows. Section 2 provides the necessary background, including a definition of dependency graphs, a discussion of different approaches to the analysis of coordination and verb groups in dependency grammar, as well as brief descriptions of PDT, MaltParser and some related work. Section 3 introduces a set of dependency graph transformations, specifically defined to deal with the dependency annotation found in PDT, which are experimentally evaluated in section 4. While the experiments reported in section 4.1 deal with pure treebank transformations, in order to establish an upper bound on what can be achieved in parsing, the experiments presented in section 4.2 examine the effects of different transformations on parsing accuracy. Finally, in section 4.3, we combine these transformations with previously proposed techniques in order to optimize overall parsing accuracy. We conclude in section 5.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 257–264,
Sydney, July 2006. ©2006 Association for Computational Linguistics

2 Background

2.1 Dependency Graphs

The basic idea in dependency parsing is that the syntactic analysis consists in establishing typed, binary relations, called *dependencies*, between the words of a sentence. This kind of analysis can be represented by a labeled directed graph, defined as follows:

- Let $R = \{r_1, \ldots, r_m\}$ be a set of dependency types (arc labels).
- A dependency graph for a string of words $W = w_1 \ldots w_n$ is a labeled directed graph $G = (W, A)$, where:
  - W is the set of nodes, i.e. word tokens in the input string, ordered by a linear precedence relation $<$.
  - A is a set of labeled arcs (w_i, r, w_j), w_i, $w_j \in W$, $r \in R$.
- A dependency graph $G = (W, A)$ is well-formed iff it is acyclic and no node has an in-degree greater than 1.

We will use the notation $w_i \xrightarrow{r} w_j$ to symbolize that $(w_i, r, w_j) \in A$, where w_i is referred to as the *head* and w_j as the *dependent*. We say that an arc is *projective* iff, for every word w_j occurring between w_i and w_k (i.e., $w_i < w_j < w_k$ or $w_i > w_j > w_k$), there is a path from w_i to w_j. A graph is projective iff all its arcs are projective. Figure 1 shows a well-formed (projective) dependency graph for a sentence from the Prague Dependency Treebank.

2.2 Coordination and Verb Groups

Dependency grammar assumes that syntactic structure consists of lexical nodes linked by binary dependencies. Dependency theories are thus best suited for binary syntactic constructions, where one element can clearly be distinguished as the syntactic head. The analysis of coordination is problematic in this respect, since it normally involves at least one conjunction and two conjuncts. The verb group, potentially consisting of a whole chain of verb forms, is another type of construction where the syntactic relation between elements is not clear-cut in dependency terms.

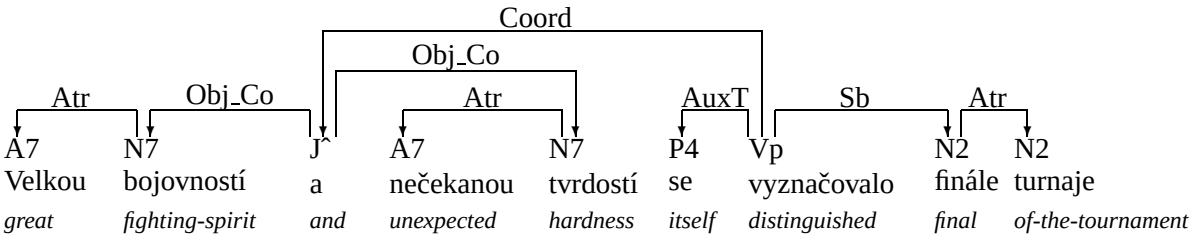
Several solutions have been proposed to the problem of coordination. One alternative is to avoid creating dependency relations between the conjuncts, and instead let the conjuncts have a direct dependency relation to the same head (Tesnière, 1959; Hudson, 1990). Another approach is to make the conjunction the head and let the conjuncts depend on the conjunction. This analysis, which appears well motivated on semantic grounds, is adopted in the FGD framework and will therefore be called Prague style (PS). It is exemplified in figure 1, where the conjunction *a* (and) is the head of the conjuncts *bojovností* and *tvrdostí*. A different solution is to adopt a more hierarchical analysis, where the conjunction depends on the first conjunct, while the second conjunct depends on the conjunction. In cases of multiple coordination, this can be generalized to a chain, where each element except the first depends on the preceding one. This more syntactically oriented approach has been advocated notably by Mel'čuk (1988) and will be called Mel'čuk style (MS). It is illustrated in figure 2, which shows a transformed version of the dependency graph in figure 1, where the elements of the coordination form a chain with the first conjunct (*bojovností*) as the topmost head. Lombardo and Lesmo (1998) conjecture that MS is more suitable than PS for incremental dependency parsing.

The difference between the more semantically oriented PS and the more syntactically oriented MS is seen also in the analysis of verb groups, where the former treats the main verb as the head, since it is the bearer of valency, while the latter treats the auxiliary verb as the head, since it is the finite element of the clause. Without questioning the theoretical validity of either approach, we can again ask which analysis is best suited to achieve high accuracy in parsing.

2.3 PDT

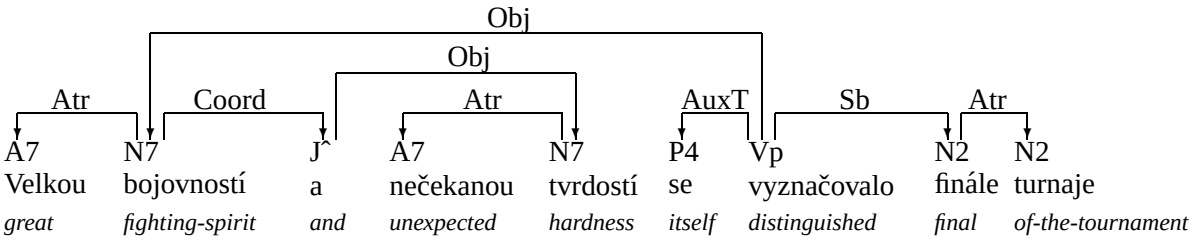
PDT (Hajič, 1998; Hajič et al., 2001) consists of 1.5M words of newspaper text, annotated in three layers: morphological, analytical and tectogrammatical. In this paper, we are only concerned with the analytical layer, which contains a surface-syntactic dependency analysis, involving a set of 28 dependency types, and not restricted to projective dependency graphs.[1] The annotation follows FGD, which means that it involves a PS analysis of both coordination and verb groups. Whether better parsing accuracy can be obtained by transforming

[1] About 2% of all dependencies are non-projective and about 25% of all sentences have a non-projective dependency graph (Nivre and Nilsson, 2005).



("The final of the tournament was distinguished by great fighting spirit and unexpected hardness")

Figure 1: Dependency graph for a Czech sentence from the Prague Dependency Treebank



("The final of the tournament was distinguished by great fighting spirit and unexpected hardness")

Figure 2: Transformed dependency graph for a Czech sentence from the Prague Dependency Treebank

this to MS is one of the hypotheses explored in the experimental study below.

2.4 MaltParser

MaltParser (Nivre and Hall, 2005; Nivre et al., 2006) is a data-driven parser-generator, which can induce a dependency parser from a treebank, and which supports several parsing algorithms and learning algorithms. In the experiments below we use the algorithm of Nivre (2003), which constructs a labeled dependency graph in one left-to-right pass over the input. Classifiers that predict the next parser action are constructed through memory-based learning (MBL), using the TiMBL software package (Daelemans and Van den Bosch, 2005), and support vector machines (SVM), using LIBSVM (Chang and Lin, 2005).

2.5 Related Work

Other ways of improving parsing accuracy with respect to coordination include learning patterns of morphological and semantical information for the conjuncts (Park and Cho, 2000). More specifically for PDT, Collins et al. (1999) relabel coordinated phrases after converting dependency structures to phrase structures, and Zeman (2004) uses a kind of pattern matching, based on frequencies of the parts-of-speech of conjuncts and conjunctions. Zeman also mentions experiments to transform the dependency structure for coordination but does not present any results.

Graph transformations in dependency parsing have also been used in order to recover non-projective dependencies together with parsers that are restricted to projective dependency graphs. Thus, Nivre and Nilsson (2005) improve parsing accuracy for MaltParser by projectivizing training data and applying an inverse transformation to the output of the parser, while Hall and Novák (2005) apply post-processing to the output of Charniak's parser (Charniak, 2000). In the final experiments below, we combine these techniques with the transformations investigated in this paper.

3 Dependency Graph Transformations

In this section, we describe algorithms for transforming dependency graphs in PDT from PS to MS and back, starting with coordination and continuing with verb groups.

3.1 Coordination

The PS-to-MS transformation for coordination will be designated $\tau_c(\Delta)$, where Δ is a data set. The transformation begins with the identification of a *base conjunction*, based on its dependency type (*Coord*) and/or its part-of-speech (*J^*). For example, the word *a* (and) in figure 1 is identified as a base conjunction.

Before the actual transformation, the base conjunction and all its dependents need to be classified into three different categories. First, the base conjunction is categorized as a *separator* (S). If the coordination consists of more than two conjuncts, it normally has one or more commas separating conjuncts, in addition to the base conjunction. These are identified by looking at their dependency type (mostly *AuxX*) and are also categorized as S. The coordination in figure 1 contains no commas, so only the word *a* will belong to S.

The remaining dependents of the base conjunction need to be divided into conjuncts (C) and other dependents (D). To make this distinction, the algorithm again looks at the dependency type. In principle, the dependency type of a conjunct has the suffix *_Co*, although special care has to be taken for coordinated prepositional cases and embedded clauses (Böhmová et al., 2003). The words *bojovností* and *tvrdostí* in figure 1, both having the dependency type *Obj_Co*, belong to the category C. Since there are no other dependents of *a*, the coordination contains no instances of the category D.

Given this classification of the words involved in a coordination, the transformation $\tau_c(\Delta)$ is straightforward and basically connects all the arcs in a chain. Let $C_1, \ldots, C_n$ be the elements of C, ordered by linear precedence, and let $S_{1_i}, \ldots, S_{m_i}$ be the separators occurring between C_i and C_{i+1}. Then every C_i becomes the head of $S_{1_i}, \ldots, S_{m_i}$, S_{m_i} becomes the head of C_{i+1}, and C_1 becomes the only dependent of the original head of the base conjunction. The dependency types of the conjuncts are truncated by removing the suffix *_Co*.[2] Also, each word in $w_d \in D$ becomes a dependent of the conjunct closest to its left, and if such a word does not exist, w_d will depend on the leftmost conjunct. After the transformation $\tau_c(\Delta)$, every coordination forms a left-headed chain, as illustrated in figure 2.

This new representation creates a problem, however. It is no longer possible to distinguish the dependents in D from other dependents of the conjuncts. For example, the word *Velkou* in figure 2 is not distinguishable from a possible dependent in D, which is an obvious drawback when transforming back to PS. One way of distinguishing D elements is to extend the set of dependency types. The dependency type r of each $w_d \in D$ can be replaced by a completely new dependency type $r+$ (e.g., $Atr+$), theoretically increasing the number of dependency types to $2 \cdot |R|$.

The inverse transformation, $\tau_c^{-1}(\Delta)$, again starts by identifying base conjunctions, using the same conditions as before. For each identified base conjunction, it calls a procedure that performs the inverse transformation by traversing the chain of conjuncts and separators "upwards" (right-to-left), collecting conjuncts (C), separators (S) and *potential* conjunction dependents (D_{pot}). When this is done, the former head of the leftmost conjunct (C_1) becomes the head of the rightmost (base) conjunction ($S_{m_{n-1}}$). In figure 2, the leftmost conjunct is *bojovností*, with the head *vyznačovalo*, and the rightmost (and only) conjunction is *a*, which will then have *vyznačovalo* as its new head. All conjuncts in the chain become dependents of the rightmost conjunction, which means that the structure is converted back to the one depicted in figure 1.

As mentioned above, the original structure in figure 1 did not have any coordination dependents, but *Velkou* $\in D_{pot}$. The last step of the inverse transformation is therefore to sort out conjunction dependents from conjunct dependents, where the former will attach to the base conjunction. Four versions have been implemented, two of which take into account the fact that the dependency types *AuxG*, *AuxX*, *AuxY*, and *Pred* are the only dependency types that are more frequent as conjunction dependents (D) than as conjunct dependents in the training data set:

- τ_c: Do not extend arc labels in τ_c. Leave all words in D_{pot} in place in τ_c^{-1}.
- τ_{c^*}: Do not extend arc labels in τ_c. Attach all words with label *AuxG*, *AuxX*, *AuxY* or *Pred* to the base conjunction in τ_c^{-1}.
- τ_{c+}: Extend arc labels from r to $r+$ for D elements in τ_c. Attach all words with label $r+$ to the base conjunction (and change the label to r) in τ_c^{-1}.
- τ_{c+^*}: Extend arc labels from r to $r+$ for D elements in τ_c, except for the labels *AuxG*, *AuxX*, *AuxY* and *Pred*. Attach all words with label $r+$, *AuxG*, *AuxX*, *AuxY*, or *Pred* to the base conjunction (and change the label to r if necessary) in τ_c^{-1}.

[2] Preliminary results indicated that this increases parsing accuracy.

3.2 Verb Groups

To transform verb groups from PS to MS, the transformation algorithm, $\tau_v(\Delta)$, starts by identifying all auxiliary verbs in a sentence. These will belong to the set A and are processed from left to right. A word $w_{aux} \in A$ iff $w_{main} \overset{AuxV}{\longrightarrow} w_{aux}$, where w_{main} is the main verb. The transformation into MS reverses the relation between the verbs, i.e., $w_{aux} \overset{AuxV}{\longrightarrow} w_{main}$, and the former head of w_{main} becomes the new head of w_{aux}. The main verb can be located on either side of the auxiliary verb and can have other dependents (whereas auxiliary verbs never have dependents), which means that dependency relations to other dependents of w_{main} may become non-projective through the transformation. To avoid this, all dependents to the left of the rightmost verb will depend on the leftmost verb, whereas the others will depend on the rightmost verb.

Performing the inverse transformation for verb groups, $\tau_v^{-1}(\Delta)$, is quite simple and essentially the same procedure inverted. Each sentence is traversed from right to left looking for arcs of the type $w_{aux} \overset{AuxV}{\longrightarrow} w_{main}$. For every such arc, the head of w_{aux} will be the new head of w_{main}, and w_{main} the new head of w_{aux}. Furthermore, since w_{aux} does not have dependents in PS, all dependents of w_{aux} in MS will become dependents of w_{main} in PS.

4 Experiments

All experiments are based on PDT 1.0, which is divided into three data sets, a training set (Δ_t), a development test set (Δ_d), and an evaluation test set (Δ_e). Table 1 shows the size of each data set, as well as the relative frequency of the specific constructions that are in focus here. Only 1.3% of all words in the training data are identified as auxiliary verbs (A), whereas coordination (S and C) is more common in PDT. This implies that coordination transformations are more likely to have a greater impact on overall accuracy compared to the verb group transformations.

In the parsing experiments reported in sections 4.1–4.2, we use Δ_t for training, Δ_d for tuning, and Δ_e for the final evaluation. The part-of-speech tagging used (both in training and testing) is the HMM tagging distributed with the treebank, with a tagging accuracy of 94.1%, and with the tagset compressed to 61 tags as in Collins et al. (1999).

Data	#S	#W	%S	%C	%A
Δ_t	73088	1256k	3.9	7.7	1.3
Δ_d	7319	126k	4.0	7.8	1.4
Δ_e	7507	126k	3.8	7.3	1.4

Table 1: PDT data sets; S = sentence, W = word; S = separator, C = conjunct, A = auxiliary verb

T	AS
τ_c	97.8
τ_{c*}	98.6
τ_{c+}	99.6
τ_{c+*}	99.4
τ_v	100.0

Table 2: Transformations; T = transformation; AS = attachment score (unlabeled) of $\tau^{-1}(\tau(\Delta_t))$ compared to Δ_t

MaltParser is used with the parsing algorithm of Nivre (2003) together with the feature model used for parsing Czech by Nivre and Nilsson (2005). In section 4.2 we use MBL, again with the same settings as Nivre and Nilsson (2005),[3] and in section 4.2 we use SVM with a polynomial kernel of degree 2.[4] The metrics for evaluation are the attachment score (AS) (labeled and unlabeled), i.e., the proportion of words that are assigned the correct head, and the exact match (EM) score (labeled and unlabeled), i.e., the proportion of sentences that are assigned a completely correct analysis. All tokens, including punctuation, are included in the evaluation scores. Statistical significance is assessed using McNemar's test.

4.1 Experiment 1: Transformations

The algorithms are fairly simple. In addition, there will always be a small proportion of syntactic constructions that do not follow the expected pattern. Hence, the transformation and inverse transformation will inevitably result in some distortion. In order to estimate the expected reduction in parsing accuracy due to this distortion, we first consider a pure treebank transformation experiment, where we compare $\tau^{-1}(\tau(\Delta_t))$ to Δ_t, for all the different transformations τ defined in the previous section. The results are shown in table 2.

We see that, even though coordination is more frequent, verb groups are easier to handle.[5] The

[3] TiMBL parameters: -k5 -mM -L3 -w0 -dID.

[4] LIBSVM parameters: -s0 -t1 -d2 -g0.12 -r0 -c1 -e0.1.

[5] The result is rounded to 100.0% but the transformed tree-

coordination version with the least loss of information (τ_{c+}) fails to recover the correct head for 0.4% of all words in Δ_t.

The difference between τ_{c+} and τ_c is expected. However, in the next section this will be contrasted with the increased burden on the parser for τ_{c+}, since it is also responsible for selecting the correct dependency type for each arc among as many as $2 \cdot |R|$ types instead of $|R|$.

4.2 Experiment 2: Parsing

Parsing experiments are carried out in four steps (for a given transformation τ):

1. Transform the training data set into $\tau(\Delta_t)$.
2. Train a parser p on $\tau(\Delta_t)$.
3. Parse a test set Δ using p with output $p(\Delta)$.
4. Transform the parser output into $\tau^{-1}(p(\Delta))$.

Table 3 presents the results for a selection of transformations using MaltParser with MBL, tested on the evaluation test set Δ_e with the untransformed data as baseline. Rows 2–5 show that transforming coordinate structures to MS improves parsing accuracy compared to the baseline, regardless of which transformation and inverse transformation are used. Moreover, the parser benefits from the verb group transformation, as seen in row 6.

The final row shows the best combination of a coordination transformation with the verb group transformation, which amounts to an improvement of roughly two percentage points, or a ten percent overall error reduction, for unlabeled accuracy.

All improvements over the baseline are statistically significant (McNemar's test) with respect to attachment score (labeled and unlabeled) and unlabeled exact match, with $p < 0.01$ except for the unlabeled exact match score of the verb group transformation, where $0.01 < p < 0.05$. For the labeled exact match, no differences are significant.

The experimental results indicate that MS is more suitable than PS as the target representation for deterministic data-driven dependency parsing. A relevant question is of course why this is the case. A partial explanation may be found in the "short-dependency preference" exhibited by most parsers (Eisner and Smith, 2005), with MaltParser being no exception. The first row of table 4 shows the accuracy of the parser for different arc lengths under the baseline condition (i.e., with no transformations). We see that it performs very well on

bank contains 19 erroneous heads.

	AS		EM	
T	**U**	**L**	**U**	**L**
None	79.08	72.83	28.99	21.15
τ_c	80.55	74.06	30.08	21.27
τ_{c^*}	80.90	74.41	30.56	21.42
τ_{c+}	80.58	74.07	30.42	21.17
τ_{c+^*}	80.87	74.36	30.89	21.38
τ_v	79.28	72.97	29.53	21.38
$\tau_v \circ \tau_{c+^*}$	81.01	74.51	31.02	21.57

Table 3: Parsing accuracy (MBL, Δ_e); T = transformation; AS = attachment score, EM = exact match; U = unlabeled, L = labeled

AS Δ_e	90.1	83.6	70.5	59.5	45.9
Length:	**1**	**2-3**	**4-6**	**7-10**	**11-**
Δ_t	51.9	**29.4**	**11.2**	**4.4**	**3.0**
$\tau_c(\Delta_t)$	**54.1**	29.1	10.7	3.8	2.4
$\tau_v(\Delta_t)$	**52.9**	29.2	10.7	4.2	2.9

Table 4: Baseline labeled AS per arc length on Δ_e (row 1); proportion of arcs per arc length in Δ_t (rows 3–5)

short arcs, but that accuracy drops quite rapidly as the arcs get longer. This can be related to the mean arc length in Δ_t, which is 2.59 in the untransformed version, 2.40 in $\tau_c(\Delta_t)$ and 2.54 in $\tau_v(\Delta_t)$. Rows 3-5 in table 4 show the distribution of arcs for different arc lengths in different versions of the data set. Both τ_c and τ_v make arcs shorter on average, which may facilitate the task for the parser.

Another possible explanation is that learning is facilitated if similar constructions are represented similarly. For instance, it is probable that learning is made more difficult when a unit has different heads depending on whether it is part of a coordination or not.

4.3 Experiment 3: Optimization

In this section we combine the best results from the previous section with the graph transformations proposed by Nivre and Nilsson (2005) to recover non-projective dependencies. We write τ_p for the projectivization of training data and τ_p^{-1} for the inverse transformation applied to the parser's output.[6] In addition, we replace MBL with SVM, a learning algorithm that tends to give higher accuracy in classifier-based parsing although it is more

[6] More precisely, we use the variant called PATH in Nivre and Nilsson (2005).

		AS		EM	
T	**LA**	**U**	**L**	**U**	**L**
None	MBL	79.08	72.83	28.99	21.15
τ_p	MBL	80.79	74.39	31.54	22.53
$\tau_p \circ \tau_v \circ \tau_{c+*}$	MBL	82.93	76.31	34.17	23.01
None	SVM	81.09	75.68	32.24	25.02
τ_p	SVM	82.93	77.28	35.99	27.05
$\tau_p \circ \tau_v \circ \tau_{c+*}$	SVM	84.55	78.82	37.63	27.69

Table 5: Optimized parsing results (SVM, Δ_e); T = transformation; LA = learning algorithm; AS = attachment score, EM = exact match; U = unlabeled, L = labeled

T	P:S	R:S	P:C	R:C	P:A	R:A	P:M	R:M
None	52.63	72.35	55.15	67.03	82.17	82.21	69.95	69.07
$\tau_p \circ \tau_v \circ \tau_{c+*}$	63.73	82.10	63.20	75.14	90.89	92.79	80.02	81.40

Table 6: Detailed results for SVM; T = transformation; P = unlabeled precision, R = unlabeled recall

costly to train (Sagae and Lavie, 2005).

Table 5 shows the results, for both MBL and SVM, of the baseline, the pure pseudo-projective parsing, and the combination of pseudo-projective parsing with PS-to-MS transformations. We see that pseudo-projective parsing brings a very consistent increase in accuracy of at least 1.5 percentage points, which is more than that reported by Nivre and Nilsson (2005), and that the addition of the PS-to-MS transformations increases accuracy with about the same margin. We also see that SVM outperforms MBL by about two percentage points across the board, and that the positive effect of the graph transformations is most pronounced for the unlabeled exact match score, where the improvement is more than five percentage points overall for both MBL and SVM.

Table 6 gives a more detailed analysis of the parsing results for SVM, comparing the optimal parser to the baseline, and considering specifically the (unlabeled) precision and recall of the categories involved in coordination (separators S and conjuncts C) and verb groups (auxiliary verbs A and main verbs M). All figures indicate, without exception, that the transformations result in higher precision and recall for all directly involved words. (All differences are significant beyond the 0.01 level.) It is worth noting that the error reduction is actually higher for A and M than for S and C, although the former are less frequent.

With respect to unlabeled attachment score, the results of the optimized parser are slightly below the best published results for a single parser. Hall and Novák (2005) report a score of 85.1%, applying a corrective model to the output of Charniak's parser; McDonald and Pereira (2006) achieve a score of 85.2% using a second-order spanning tree algorithm. Using ensemble methods and a pool of different parsers, Zeman and Žabokrtský (2005) attain a top score of 87.0%. For unlabeled exact match, our results are better than any previously reported results, including those of McDonald and Pereira (2006). (For the labeled scores, we are not aware of any comparable results in the literature.)

5 Conclusion

The results presented in this paper confirm that choosing the right representation is important in parsing. By systematically transforming the representation of coordinate structures and verb groups in PDT, we achieve a 10% error reduction for a data-driven dependency parser. Adding graph transformations for non-projective dependency parsing gives a total error reduction of about 20% (even more for unlabeled exact match). In this way, we achieve state-of-the-art accuracy with a deterministic, classifier-based dependency parser.

Acknowledgements

The research presented in this paper was partially supported by the Swedish Research Council. We are grateful to Jan Hajič and Daniel Zeman for help with the Czech data and to three anonymous reviewers for helpful comments and suggestions.

References

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30:479–511.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague Dependency Treebank: A three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers.

Chih-Chung Chang and Chih-Jen Lin. 2005. LIBSVM: A library for support vector machines.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 132–139.

Michael Collins, Jan Hajič, Eric Brill, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 505–512.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annatual Meeting of the Association for Computational Linguistics (ACL)*, pages 16–23.

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

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

Jason Eisner and Noah A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*.

Jan Hajič, Barbora Vidova Hladka, Jarmila Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. 2001. Prague Dependency Treebank 1.0. LDC, 2001T10.

Jan Hajič. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning*, pages 12–19. Prague Karolinum, Charles University Press.

Keith Hall and Vaclav Novák. 2005. Corrective modeling for non-projective dependency parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*.

Richard Hudson. 1990. *English Word Grammar*. Basil Blackwell.

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

Dan Klein and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 423–430.

Vincenzo Lombardo and Leonardo Lesmo. 1998. Unit coordination and gapping in dependency theory. In *Proceedings of the Workshop on Processing of Dependency-Based Grammars*, pages 11–20.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Igor Mel'cuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Joakim Nivre and Johan Hall. 2005. MaltParser: A language-independent system for data-driven dependency parsing. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT)*, pages 137–148.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Jong C. Park and Hyung Joon Cho. 2000. Informed parsing for coordination with combinatory categorial grammar. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 593–599.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pages 125–132.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Pragmatic Aspects*. Reidel.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Editions Klincksieck.

Daniel Zeman and Zdeněk Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*.

Daniel Zeman. 2004. *Parsing with a Statistical Dependency Model*. Ph.D. thesis, Charles University.