Generalizing Tree Transformations for Inductive Dependency Parsing

Jens Nilsson* **Joakim Nivre*†** **Johan Hall***

*Växjö University, School of Mathematics and Systems Engineering, Sweden
†Uppsala University, Dept. of Linguistics and Philology, Sweden
{jni,nivre,jha}@msi.vxu.se

Abstract

Previous studies in data-driven dependency parsing have shown that tree transformations can improve parsing accuracy for specific parsers and data sets. We investigate to what extent this can be generalized across languages/treebanks and parsers, focusing on pseudo-projective parsing, as a way of capturing non-projective dependencies, and transformations used to facilitate parsing of coordinate structures and verb groups. The results indicate that the beneficial effect of pseudo-projective parsing is independent of parsing strategy but sensitive to language or treebank specific properties. By contrast, the construction specific transformations appear to be more sensitive to parsing strategy but have a constant positive effect over several languages.

1 Introduction

Treebank parsers are trained on syntactically annotated sentences and a major part of their success can be attributed to extensive manipulations of the training data as well as the output of the parser, usually in the form of various tree transformations. This can be seen in state-of-the-art constituency-based parsers such as Collins (1999), Charniak (2000), and Petrov et al. (2006), and the effects of different transformations have been studied by Johnson (1998), Klein and Manning (2003), and Bikel (2004). Corresponding manipulations in the form of tree transformations for dependency-based parsers have recently gained more interest (Nivre and Nilsson, 2005; Hall and Novák, 2005; McDonald and Pereira, 2006; Nilsson et al., 2006) but are still less studied, partly because constituency-based parsing has dominated the field for a long time, and partly because dependency structures have less structure to manipulate than constituent structures.

Most of the studies in this tradition focus on a particular parsing model and a particular data set, which means that it is difficult to say whether the effect of a given transformation is dependent on a particular parsing strategy or on properties of a particular language or treebank, or both. The aim of this study is to further investigate some tree transformation techniques previously proposed for data-driven dependency parsing, with the specific aim of trying to generalize results across languages/treebanks and parsers. More precisely, we want to establish, first of all, whether the transformation as such makes specific assumptions about the language, treebank or parser and, secondly, whether the improved parsing accuracy that is due to a given transformation is constant across different languages, treebanks, and parsers.

The three types of syntactic phenomena that will be studied here are non-projectivity, coordination and verb groups, which in different ways pose problems for dependency parsers. We will focus on tree transformations that combine preprocessing with post-processing, and where the parser is treated as a black box, such as the pseudo-projective parsing technique proposed by Nivre and Nilsson (2005) and the tree transformations investigated in Nilsson et al. (2006). To study the influence of lan-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 968–975,
Prague, Czech Republic, June 2007.

guage and treebank specific properties we will use data from Arabic, Czech, Dutch, and Slovene, taken from the CoNLL-X shared task on multilingual dependency parsing (Buchholz and Marsi, 2006). To study the influence of parsing methodology, we will compare two different parsers: MaltParser (Nivre et al., 2004) and MSTParser (McDonald et al., 2005). Note that, while it is possible in principle to distinguish between syntactic properties of a language as such and properties of a particular syntactic annotation of the language in question, it will be impossible to tease these apart in the experiments reported here, since this would require having not only multiple languages but also multiple treebanks for each language. In the following, we will therefore speak about the properties of *treebanks* (rather than *languages*), but it should be understood that these properties in general depend both on properties of the language and of the particular syntactic annotation adopted in the treebank.

The rest of the paper is structured as follows. Section 2 surveys tree transformations used in dependency parsing and discusses dependencies between transformations, on the one hand, and treebanks and parsers, on the other. Section 3 introduces the four treebanks used in this study, and section 4 briefly describes the two parsers. Experimental results are presented in section 5 and conclusions in section 6.

2 Background

2.1 Non-projectivity

The tree transformations that have attracted most interest in the literature on dependency parsing are those concerned with recovering non-projectivity. The definition of non-projectivity can be found in Kahane et al. (1998). Informally, an arc is *projective* if all tokens it covers are descendants of the arc's head token, and a dependency tree is projective if all its arcs are projective.[1]

The full potential of dependency parsing can only be realized if non-projectivity is allowed, which pose a problem for projective dependency parsers. Direct non-projective parsing can be performed with good accuracy, e.g., using the Chu-Liu-Edmonds algorithm, as proposed by McDonald et al. (2005). On the other hand, non-projective parsers tend, among other things, to be slower. In order to maintain the benefits of projective parsing, tree transformations techniques to recover non-projectivity while using a projective parser have been proposed in several studies, some described below.

In discussing the recovery of empty categories in data-driven constituency parsing, Campbell (2004) distinguishes between approaches based on pure post-processing and approaches based on a combination of preprocessing and post-processing. The same division can be made for the recovery of non-projective dependencies in data-driven dependency parsing.

Pure Post-processing

Hall and Novák (2005) propose a corrective modeling approach. The motivation is that the parsers of Collins et al. (1999) and Charniak (2000) adapted to Czech are not able to create the non-projective arcs present in the treebank, which is unsatisfactory. They therefore aim to correct erroneous arcs in the parser's output (specifically all those arcs which should be non-projective) by training a classifier that predicts the most probable head of a token in the neighborhood of the head assigned by the parser.

Another example is the second-order approximate spanning tree parser developed by McDonald and Pereira (2006). It starts by producing the highest scoring projective dependency tree using Eisner's algorithm. In the second phase, tree transformations are performed, replacing lower scoring projective arcs with higher scoring non-projective ones.

Preprocessing with Post-processing

The training data can also be preprocessed to facilitate the recovery of non-projective arcs in the output of a projective parser. The pseudo-projective transformation proposed by Nivre and Nilsson (2005) is such an approach, which is compatible with different parser engines.

First, the training data is *projectivized* by making non-projective arcs projective using a lifting operation. This is combined with an augmentation of the dependency labels of projectivized arcs (and/or surrounding arcs) with information that probably reveals their correct non-projective positions. The out-

[1] If dependency arcs are drawn above the linearly ordered sequence of tokens, preceded by a special root node, then a non-projective dependency tree always has crossing arcs.

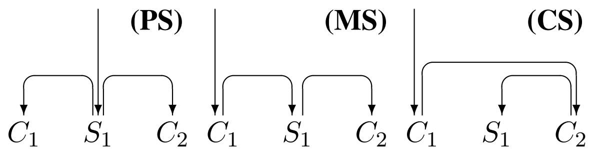

Figure 1: Dependency structure for coordination

put of the parser, trained on the projectivized data, is then *deprojectivized* by a heuristic search using the added information in the dependency labels. The only assumption made about the parser is therefore that it can learn to derive labeled dependency structures with augmented dependency labels.

2.2 Coordination and Verb Groups

The second type of transformation concerns linguistic phenomena that are not impossible for a projective parser to process but which may be difficult to learn, given a certain choice of dependency analysis. This study is concerned with two such phenomena, coordination and verb groups, for which tree transformations have been shown to improve parsing accuracy for MaltParser on Czech (Nilsson et al., 2006). The general conclusion of this study is that coordination and verb groups in the Prague Dependency Treebank (PDT), based on theories of the Prague school (PS), are annotated in a way that is difficult for the parser to learn. By transforming coordination and verb groups in the training data to an annotation similar to that advocated by Mel'čuk (1988) and then performing an inverse transformation on the parser output, parsing accuracy can therefore be improved. This is again an instance of the black-box idea.

Schematically, coordination is annotated in the Prague school as depicted in PS in figure 1, where the conjuncts are dependents of the conjunction. In Mel'čuk style (MS), on the other hand, conjuncts and conjunction(s) form a chain going from left to right. A third way of treating coordination, not discussed by Nilsson et al. (2006), is used by the parser of Collins (1999), which internally represents coordination as a direct relation between the conjuncts. This is illustrated in CS in figure 1, where the conjunction depends on one of the conjuncts, in this case on the rightmost one.

Nilsson et al. (2006) also show that the annotation of verb groups is not well-suited for parsing PDT using MaltParser, and that transforming the dependency structure for verb groups has a positive impact on parsing accuracy. In PDT, auxiliary verbs are dependents of the main verb, whereas it according to Mel'čuk is the (finite) auxiliary verb that is the head of the main verb. Again, the parsing experiments in this study show that verb groups are more difficult to parse in PS than in MS.

2.3 Transformations, Parsers, and Treebanks

Pseudo-projective parsing and transformations for coordination and verb groups are instances of the same general methodology:

1. Apply a tree transformation to the training data.
2. Train a parser on the transformed data.
3. Parse new sentences.
4. Apply an inverse transformation to the output of the parser.

In this scheme, the parser is treated as a black box. All that is assumed is that it is a data-driven parser designed for (projective) labeled dependency structures. In this sense, the tree transformations are independent of parsing methodology. Whether the beneficial effect of a transformation, if any, is also independent of parsing methodology is another question, which will be addressed in the experimental part of this paper.

The pseudo-projective transformation is independent not only of parsing methodology but also of treebank (and language) specific properties, as long as the target representation is a (potentially non-projective) labeled dependency structure. By contrast, the coordination and verb group transformations presuppose not only that the language in question contains these constructions but also that the treebank adopts a PS annotation. In this sense, they are more limited in their applicability than pseudo-projective parsing. Again, it is a different question whether the transformations have a positive effect for all treebanks (languages) to which they can be applied.

3 Treebanks

The experiments are mostly conducted using treebank data from the CoNLL shared task 2006. This

	Slovene SDT	Arabic PADT	Dutch Alpino	Czech PDT
# T	29	54	195	1249
# S	1.5	1.5	13.3	72.7
%-NPS	22.2	11.2	36.4	23.2
%-NPA	1.8	0.4	5.4	1.9
%-C	9.3	8.5	4.0	8.5
%-A	8.8	-	-	1.3

Table 1: Overview of the data sets (ordered by size), where # S * 1000 = number of sentences, # T * 1000 = number of tokens, %-NPS = percentage of non-projective sentences, %-NPA = percentage of non-projective arcs, %-C = percentage of conjuncts, %-A = percentage of auxiliary verbs.

subsection summarizes some of the important characteristics of these data sets, with an overview in table 1. Any details concerning the conversion from the original formats of the various treebanks to the CoNLL format, a pure dependency based format, are found in documentation referred to in Buchholz and Marsi (2006).

PDT (Hajič et al., 2001) is the largest manually annotated treebank, and as already mentioned, it adopts PS for coordination and verb groups. As the last four rows reveal, PDT contains a quite high proportion of non-projectivity, since almost every fourth dependency graph contains at least one non-projective arc. The table also shows that coordination is more common than verb groups in PDT. Only 1.3% of the tokens in the training data are identified as auxiliary verbs, whereas 8.5% of the tokens are identified as conjuncts.

Both Slovene Dependency Treebank (Džeroski et al., 2006) (SDT) and Prague Arabic Dependency Treebank (Hajič et al., 2004) (PADT) annotate coordination and verb groups as in PDT, since they too are influenced by the theories of the Prague school. The proportions of non-projectivity and conjuncts in SDT are in fact quite similar to the proportions in PDT. The big difference is the proportion of auxiliary verbs, with many more auxiliary verbs in SDT than in PDT. It is therefore plausible that the transformations for verb groups will have a larger impact on parser accuracy in SDT.

Arabic is not a Slavic languages such as Czech and Slovene, and the annotation in PADT is therefore more dissimilar to PDT than SDT is. One such example is that Arabic does not have auxiliary verbs. Table 1 thus does not give figures verb groups. The amount of coordination is on the other hand comparable to both PDT and SDT. The table also reveals that the amount of non-projective arcs is about 25% of that in PDT and SDT, although the amount of non-projective sentences is still as large as 50% of that in PDT and SDT.

Alpino (van der Beek et al., 2002) in the CoNLL format, the second largest treebank in this study, is not as closely tied to the theories of the Prague school as the others, but still treats coordination in a way similar to PS. The table shows that coordination is less frequent in the CoNLL version of Alpino than in the three other treebanks. The other characteristic of Alpino is the high share of non-projectivity, where more than every third sentence is non-projective. Finally, the lack of information about the share of auxiliary verbs is not due to the non-existence of such verbs in Dutch but to the fact that Alpino adopts an MS annotation of verb groups (i.e., treating main verbs as dependents of auxiliary verbs), which means that the verb group transformation of Nilsson et al. (2006) is not applicable.

4 Parsers

The parsers used in the experiments are MaltParser (Nivre et al., 2004) and MSTParser (McDonald et al., 2005). These parsers are based on very different parsing strategies, which makes them suitable in order to test the parser independence of different transformations. MaltParser adopts a greedy, deterministic parsing strategy, deriving a labeled dependency structure in a single left-to-right pass over the input and uses support vector machines to predict the next parsing action. MSTParser instead extracts a maximum spanning tree from a dense weighted graph containing all possible dependency arcs between tokens (with Eisner's algorithm for projective dependency structures or the Chu-Liu-Edmonds algorithm for non-projective structures), using a global discriminative model and online learning to assign weights to individual arcs.[2]

[2]The experiments in this paper are based on the first-order factorization described in McDonald et al. (2005)

5 Experiments

The experiments reported in section 5.1–5.2 below are based on the training sets from the CoNLL-X shared task, except where noted. The results reported are obtained by a ten-fold cross-validation (with a pseudo-randomized split) for all treebanks except PDT, where 80% of the data was used for training and 20% for development testing (again with a pseudo-randomized split). In section 5.3, we give results for the final evaluation on the CoNLL-X test sets using all three transformations together with MaltParser.

Parsing accuracy is primarily measured by the unlabeled attachment score (AS$_U$), i.e., the proportion of tokens that are assigned the correct head, as computed by the official CoNLL-X evaluation script with default settings (thus excluding all punctuation tokens). In section 5.3 we also include the *labeled* attachment score (AS$_L$) (where a token must have both the correct head and the correct dependency label to be counted as correct), which was the official evaluation metric in the CoNLL-X shared task.

5.1 Comparing Treebanks

We start by examining the effect of transformations on data from different treebanks (languages), using a single parser: MaltParser.

Non-projectivity

The question in focus here is whether the effect of the pseudo-projective transformation for MaltParser varies with the treebank. Table 2 presents the unlabeled attachment score results (AS_U), comparing the pseudo-projective parsing technique (P-Proj) with two baselines, obtained by training the strictly projective parser on the original (non-projective) training data (N-Proj) and on projectivized training data with no augmentation of dependency labels (Proj).

The first thing to note is that pseudo-projective parsing gives a significant improvement for PDT, as previously reported by Nivre and Nilsson (2005), but also for Alpino, where the improvement is even larger, presumably because of the higher proportion of non-projective dependencies in the Dutch treebank. By contrast, there is no significant improvement for either SDT or PADT, and even a small drop in the accuracy figures for SDT. Finally, in contrast to the results reported by Nivre and Nilsson (2005), simply projectivizing the training data (without using an inverse transformation) is not beneficial at all, except possibly for Alpino.

	N-Proj	**Proj**	**P-Proj**
SDT	**77.27**	76.63**	77.11
PADT	76.96	**77.07***	**77.07***
Alpino	82.75	83.28**	**87.08****
PDT	83.41	83.32**	**84.42****

Table 2: AS$_U$ for pseudo-projective parsing with MaltParser. McNemar's test: $^{*} = p < .05$ and $^{**} = p < 0.01$ compared to **N-Proj**.

	1	**2**	**3**	**>3**
SDT	88.4	9.1	1.7	0.84
PADT	66.5	14.4	5.2	13.9
Alpino	84.6	13.8	1.5	0.07
PDT	93.8	5.6	0.5	0.1

Table 3: The number of lifts for non-projective arcs.

But why does not pseudo-projective parsing improve accuracy for SDT and PADT? One possible factor is the complexity of the non-projective constructions, which can be measured by counting the number of lifts that are required to make non-projective arcs projective. The more deeply nested a non-projective arc is, the more difficult it is to recover because of parsing errors as well as search errors in the inverse transformation. The figures in table 3 shed some interesting light on this factor.

For example, whereas 93.8% of all arcs in PDT require only one lift before they become projective (88.4% and 84.6% for SDT and Alpino, respectively), the corresponding figure for PADT is as low as 66.5%. PADT also has a high proportion of very deeply nested non-projective arcs (>3) in comparison to the other treebanks, making the inverse transformation for PADT more problematic than for the other treebanks. The absence of a positive effect for PADT is therefore understandable given the deeply nested non-projective constructions in PADT.

However, one question that still remains is why SDT and PDT, which are so similar in terms of both nesting depth and amount of non-projectivity, be-

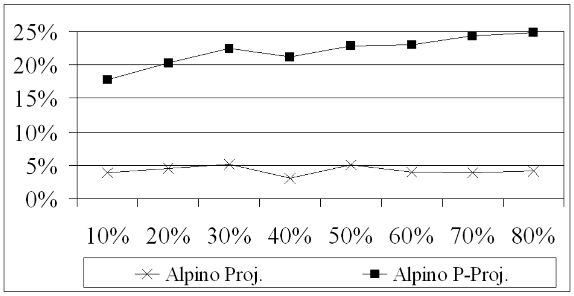

Figure 2: Learning curves for Alpino measured as error reduction for AS_U.

have differently with respect to pseudo-projective parsing. Another factor that may be important here is the amount of training data available. As shown in table 1, PDT is more than 40 times larger than SDT. To investigate the influence of training set size, a learning curve experiment has been performed. Alpino is a suitable data set for this due to its relatively large amount of both data and non-projectivity.

Figure 2 shows the learning curve for pseudo-projective parsing (P-Proj), compared to using only projectivized training data (Proj), measured as error reduction in relation to the original non-projective training data (N-Proj). The experiment was performed by incrementally adding cross-validation folds 1–8 to the training set, using folds 9–0 as static test data.

One can note that the error reduction for Proj is unaffected by the amount of data. While the error reduction varies slightly, it turns out that the error reduction is virtually the same for 10% of the training data as for 80%. That is, there is no correlation if information concerning the lifts are not added to the labels. However, with a pseudo-projective transformation, which actively tries to recover non-projectivity, the learning curve clearly indicates that the amount of data matters. Alpino, with 36% non-projective sentences, starts at about 17% and has a climbing curve up to almost 25%.

Although this experiment shows that there is a correlation between the amount of data and the accuracy for pseudo-projective parsing, it does probably not tell the whole story. If it did, one would expect that the error reduction for the pseudo-projective transformation would be much closer to Proj when

	None	**Coord**	**VG**
SDT	77.27	79.33**	77.92**
PADT	76.96	79.05**	-
Alpino	82.75	83.38**	-
PDT	83.41	85.51**	83.58**

Table 4: AS_U for coordination and verb group transformations with MaltParser (None = N-Proj). McNemar's test: $^{**} = p < .01$ compared to **None**.

the amount of data is low (to the left in the figure) than they apparently are. Of course, the difference is likely to diminish with even less data, but it should be noted that 10% of Alpino has about half the size of PADT, for which the positive impact of pseudo-projective parsing is absent. The absence of increased accuracy for SDT can partially be explained by the higher share of non-projective arcs in Alpino (3 times more).

Coordination and Verb Groups

The corresponding parsing results using MaltParser with transformations for coordination and verb groups are shown in table 4. For SDT, PADT and PDT, the annotation of coordination has been transformed from PS to MS, as described in Nilsson et al. (2006). For Alpino, the transformation is from PS to CS (cf. section 2.2), which was found to give slightly better performance in preliminary experiments. The baseline with no transformation (None) is the same as N-Proj in table 2.

As the figures indicate, transforming coordination is beneficial not only for PDT, as reported by Nilsson et al. (2006), but also for SDT, PADT, and Alpino. It is interesting to note that SDT, PADT and PDT, with comparable amounts of conjuncts, have comparable increases in accuracy (about 2 percentage points each), despite the large differences in training set size. It is therefore not surprising that Alpino, with a much smaller amount of conjuncts, has a lower increase in accuracy. Taken together, these results indicate that the frequency of the construction is more important than the size of the training set for this type of transformation.

The same generalization over treebanks holds for verb groups too. The last column in table 4 shows that the expected increase in accuracy for PDT is ac-

Algorithm	N-Proj	Proj	P-Proj
Eisner	81.79	83.23	**86.45**
CLE	86.39		

Table 5: Pseudo-projective parsing results (AS_U) for Alpino with MSTParser.

companied by a even higher increase for SDT. This can probably be attributed to the higher frequency of auxiliary verbs in SDT.

5.2 Comparing Parsers

The main question in this section is to what extent the positive effect of different tree transformations is dependent on parsing strategy, since all previous experiments have been performed with a single parser (MaltParser). For comparison we have performed two experiments with MSTParser, version 0.1, which is based on a very different parsing methodology (cf. section 4). Due to some technical difficulties (notably the very high memory consumption when using MSTParser for labeled dependency parsing), we have not been able to replicate the experiments from the preceding section exactly. The results presented below must therefore be regarded as a preliminary exploration of the dependencies between tree transformations and parsing strategy.

Table 5 presents AS_U results for MSTParser in combination with pseudo-projective parsing applied to the Alpino treebank of Dutch.[3] The first row contains the result for Eisner's algorithm using no transformation (N-Proj), projectivized training data (Proj), and pseudo-projective parsing (P-Proj). The figures show a pattern very similar to that for MaltParser, with a boost in accuracy for Proj compared to N-Proj, and with a significantly higher accuracy for P-Proj over Proj. It is also worth noting that the error reduction between N-Proj and P-Proj is actually higher for MSTParser here than for MaltParser in table 2.

The second row contains the result for the Chu-Liu-Edmonds algorithm (CLE), which constructs non-projective structures directly and therefore does not require the pseudo-projective transformation. A comparison between Eisner's algorithm with pseudo-projective transformation and CLE reveals that pseudo-projective parsing is at least as accurate as non-projective parsing for AS_U. (The small difference is not statistically significant.)

[3]The figures are not completely comparable to the previously presented Dutch results for MaltParser, since MaltParser's feature model has access to all the information in the CoNLL data format, whereas MSTParser in this experiment only could handle word forms and part-of-speech tags.

Trans.	None	Coord	VG
AS_U	84.5	83.5	84.5

Table 6: Coordination and verb group transformations for PDT with the CLE algorithm.

		Dev	Eval	Niv	McD
SDT	AS_U	80.40	82.01	78.72	83.17
	AS_L	71.06	72.44	70.30	73.44
PADT	AS_U	78.97	78.56	77.52	79.34
	AS_L	67.63	67.58	66.71	66.91
Alpino	AS_U	87.63	82.85	81.35	83.57
	AS_L	84.02	79.73	78.59	79.19
PDT	AS_U	85.72	85.98	84.80	87.30
	AS_L	78.56	78.80	78.42	80.18

Table 7: Evaluation on CoNLL-X test data; MaltParser with all transformations (**Dev** = development, **Eval** = CoNLL test set, **Niv** = Nivre et al. (2006), **McD** = McDonald et al. (2006))

By contrast, no positive effect could be detected for the coordination and verb group transformations togther with MSTParser. The figures in table 6 are not based on CoNLL data, but instead on the evaluation test set of the original PDT 1.0, which enables a direct comparison to McDonald et. al. (2005) (the **None** column). We see that there is even a negative effect for the coordination transformation. These results clearly indicate that the effect of these transformations is at least partly dependent on parsing strategy, in contrast to what was found for the pseudo-projective parsing technique.

5.3 Combining Transformations

In order to assess the combined effect of all three transformations in relation to the state of the art, we performed a final evaluation using MaltParser on the dedicated test sets from the CoNLL-X shared task. Table 7 gives the results for both development (cross-validation for SDT, PADT, and Alpino;

development set for PDT) and final test, compared to the two top performing systems in the shared task, MSTParser with approximate second-order non-projective parsing (McDonald et al., 2006) and MaltParser with pseudo-projective parsing (but no coordination or verb group transformations) (Nivre et al., 2006). Looking at the labeled attachment score (AS_L), the official scoring metric of the CoNLL-X shared task, we see that the combined effect of the three transformations boosts the performance of MaltParser for all treebanks and in two cases out of four outperforms MSTParser (which was the top scoring system for all four treebanks).

6 Conclusion

In this paper, we have examined the generality of tree transformations for data-driven dependency parsing. The results indicate that the pseudo-projective parsing technique has a positive effect on parsing accuracy that is independent of parsing methodology but sensitive to the amount of training data as well as to the complexity of non-projective constructions. By contrast, the construction-specific transformations targeting coordination and verb groups appear to have a more language-independent effect (for languages to which they are applicable) but do not help for all parsers. More research is needed in order to know exactly what the dependencies are between parsing strategy and tree transformations. Regardless of this, however, it is safe to conclude that pre-processing and post-processing is important not only in constituency-based parsing, as previously shown in a number of studies, but also for inductive dependency parsing.

References

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

S. Buchholz and E. Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proceedings of CoNLL*, pages 1–17.

R. Campbell. 2004. Using Linguistic Principles to Recover Empty Categories. In *Proceedings of ACL*, pages 645–652.

E. Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of NAACL*, pages 132–139.

M. Collins, J. Hajič, L. Ramshaw, and C. Tillmann. 1999. A statistical parser for Czech. In *Proceedings of ACL*, pages 100–110.

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

S. Džeroski, T. Erjavec, N. Ledinek, P. Pajas, Z. Žabokrtsky, and A. Žele. 2006. Towards a Slovene Dependency Treebank. In *LREC*.

J. Hajič, B. V. Hladka, J. Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. 2001. Prague Dependency Treebank 1.0. LDC, 2001T10.

J. Hajič, O. Smrž, P. Zemánek, J. Šnaidauf, and E. Beška. 2004. Prague Arabic Dependency Treebank: Development in Data and Tools. In *NEMLAR*, pages 110–117.

K. Hall and V. Novák. 2005. Corrective modeling for non-projective dependency parsing. In *Proceedings of IWPT*, pages 42–52.

M. Johnson. 1998. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24:613–632.

S. Kahane, A. Nasr, and O. Rambow. 1998. Pseudo-Projectivity: A Polynomially Parsable Non-Projective Dependency Grammar. In *Proceedings of COLING/ACL*, pages 646–652.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430.

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

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP*, pages 523–530.

R. McDonald, K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL*, pages 216–220.

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

J. Nilsson, J. Nivre, and J. Hall. 2006. Graph Transformations in Data-Driven Dependency Parsing. In *Proceedings of COLING/ACL*, pages 257–264.

J. Nivre and J. Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of ACL*, pages 99–106.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based Dependency Parsing. In H. T. Ng and E. Riloff, editors, *Proceedings of CoNLL*, pages 49–56.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006. Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines. In *Proceedings of CoNLL*, pages 221–225.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of COLING/ACL*, pages 433–440.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. In *Computational Linguistics in the Netherlands (CLIN)*.