Deterministic Dependency Parsing of English Text

Joakim Nivre and **Mario Scholz**
School of Mathematics and Systems Engineering
Växjö University
SE-35195 Växjö
Sweden
joakim.nivre@msi.vxu.se

Abstract

This paper presents a deterministic dependency parser based on memory-based learning, which parses English text in linear time. When trained and evaluated on the Wall Street Journal section of the Penn Treebank, the parser achieves a maximum attachment score of 87.1%. Unlike most previous systems, the parser produces labeled dependency graphs, using as arc labels a combination of bracket labels and grammatical role labels taken from the Penn Treebank II annotation scheme. The best overall accuracy obtained for identifying both the correct head and the correct arc label is 86.0%, when restricted to grammatical role labels (7 labels), and 84.4% for the maximum set (50 labels).

1 Introduction

There has been a steadily increasing interest in syntactic parsing based on dependency analysis in recent years. One important reason seems to be that dependency parsing offers a good compromise between the conflicting demands of analysis depth, on the one hand, and robustness and efficiency, on the other. Thus, whereas a complete dependency structure provides a fully disambiguated analysis of a sentence, this analysis is typically less complex than in frameworks based on constituent analysis and can therefore often be computed deterministically with reasonable accuracy. Deterministic methods for dependency parsing have now been applied to a variety of languages, including Japanese (Kudo and Matsumoto, 2000), English (Yamada and Matsumoto, 2003), Turkish (Oflazer, 2003), and Swedish (Nivre et al., 2004).

For English, the interest in dependency parsing has been weaker than for other languages. To some extent, this can probably be explained by the strong tradition of constituent analysis in Anglo-American linguistics, but this trend has been reinforced by the fact that the major treebank of American English, the Penn Treebank (Marcus et al., 1993), is annotated primarily with constituent analysis. On the other hand, the best available parsers trained on the Penn Treebank, those of Collins (1997) and Charniak (2000), use statistical models for disambiguation that make crucial use of dependency relations. Moreover, the deterministic dependency parser of Yamada and Matsumoto (2003), when trained on the Penn Treebank, gives a dependency accuracy that is almost as good as that of Collins (1997) and Charniak (2000).

The parser described in this paper is similar to that of Yamada and Matsumoto (2003) in that it uses a deterministic parsing algorithm in combination with a classifier induced from a treebank. However, there are also important differences between the two approaches. First of all, whereas Yamada and Matsumoto employs a strict bottom-up algorithm (essentially shift-reduce parsing) with multiple passes over the input, the present parser uses the algorithm proposed in Nivre (2003), which combines bottom-up and top-down processing in a single pass in order to achieve incrementality. This also means that the time complexity of the algorithm used here is linear in the size of the input, while the algorithm of Yamada and Matsumoto is quadratic in the worst case. Another difference is that Yamada and Matsumoto use support vector machines (Vapnik, 1995), while we instead rely on memory-based learning (Daelemans, 1999).

Most importantly, however, the parser presented in this paper constructs labeled dependency graphs, i.e. dependency graphs where arcs are labeled with dependency types. As far as we know, this makes it different from all previous systems for dependency parsing applied to the Penn Treebank (Eisner, 1996; Yamada and Matsumoto, 2003), although there are systems that extract labeled grammatical relations based on shallow parsing, e.g. Buchholz (2002). The fact that we are working with labeled dependency graphs is also one of the motivations for choosing memory-based learning over support vector machines, since we require a multi-class classifier. Even though it is possible to use SVM for multi-class classification, this can get cumbersome when the number of classes is large. (For the

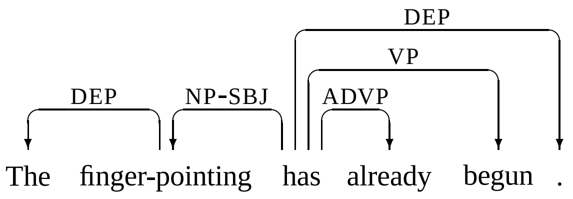


Figure 1: Dependency graph for English sentence

unlabeled dependency parser of Yamada and Matsumoto (2003) the classification problem only involves three classes.)

The parsing methodology investigated here has previously been applied to Swedish, where promising results were obtained with a relatively small treebank (approximately 5000 sentences for training), resulting in an attachment score of 84.7% and a labeled accuracy of 80.6% (Nivre et al., 2004).[1] However, since there are no comparable results available for Swedish, it is difficult to assess the significance of these findings, which is one of the reasons why we want to apply the method to a benchmark corpus such as the the Penn Treebank, even though the annotation in this corpus is not ideal for labeled dependency parsing.

The paper is structured as follows. Section 2 describes the parsing algorithm, while section 3 explains how memory-based learning is used to guide the parser. Experimental results are reported in section 4, and conclusions are stated in section 5.

2 Deterministic Dependency Parsing

In dependency parsing the goal of the parsing process is to construct a labeled dependency graph of the kind depicted in Figure 1. In formal terms, we define dependency graphs as follows:

1. Let $R = \{r_1, \ldots, r_m\}$ be the set of permissible dependency types (arc labels).
2. A dependency graph for a string of words $W = w_1 \cdots w_n$ is a labeled directed graph $D = (W, A)$, where
   (a) W is the set of nodes, i.e. word tokens in the input string,
   (b) A is a set of labeled arcs (w_i, r, w_j) ($w_i, w_j \in W$, $r \in R$),
   (c) for every $w_j \in W$, there is at most one arc $(w_i, r, w_j) \in A$.
3. A graph $D = (W, A)$ is well-formed iff it is acyclic, projective and connected.

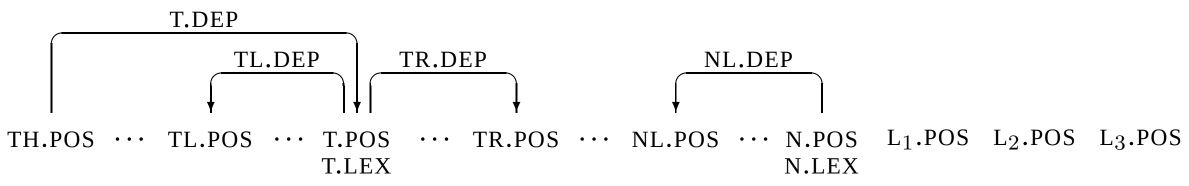
[1]The attachment score only considers whether a word is assigned the correct head; the labeled accuracy score in addition requires that it is assigned the correct dependency type; cf. section 4.

For a more detailed discussion of dependency graphs and well-formedness conditions, the reader is referred to Nivre (2003).

The parsing algorithm used here was first defined for unlabeled dependency parsing in Nivre (2003) and subsequently extended to labeled graphs in Nivre et al. (2004). Parser configurations are represented by triples $\langle S, I, A \rangle$, where S is the stack (represented as a list), I is the list of (remaining) input tokens, and A is the (current) arc relation for the dependency graph. (Since in a dependency graph the set of nodes is given by the input tokens, only the arcs need to be represented explicitly.) Given an input string W, the parser is initialized to $\langle \mathbf{nil}, W, \emptyset \rangle$[2] and terminates when it reaches a configuration $\langle S, \mathbf{nil}, A \rangle$ (for any list S and set of arcs A). The input string W is *accepted* if the dependency graph $D = (W, A)$ given at termination is well-formed; otherwise W is *rejected*. Given an arbitrary configuration of the parser, there are four possible transitions to the next configuration (where t is the token on top of the stack, n is the next input token, w is any word, and $r, r' \in R$):

1. **Left-Arc:** In a configuration $\langle t|S,n|I,A \rangle$, if there is no arc $(w, r, t) \in A$, extend A with (n, r', t) and pop the stack, giving the configuration $\langle S,n|I,A \cup \{(n, r', t)\} \rangle$.
2. **Right-Arc**: In a configuration $\langle t|S,n|I,A \rangle$, if there is no arc $(w, r, n) \in A$, extend A with (t, r', n) and push n onto the stack, giving the configuration $\langle n|t|S,I,A \cup \{(t, r', n)\} \rangle$.
3. **Reduce**: In a configuration $\langle t|S,I,A \rangle$, if there is an arc $(w, r, t) \in A$, pop the stack, giving the configuration $\langle S,I,A \rangle$.
4. **Shift:** In a configuration $\langle S,n|I,A \rangle$, push n onto the stack, giving the configuration $\langle n|S,I,A \rangle$.

[2]We use **nil** to denote the empty list and $a|A$ to denote a list with head a and tail A.


T.DEP
TL.DEP
TR.DEP
NL.DEP
TH.POS ··· TL.POS ··· T.POS ··· TR.POS ··· NL.POS ··· N.POS L1.POS L2.POS L3.POS
T.LEX
N.LEX

T	=	Top of the stack
N	=	Next input token
TL	=	Leftmost dependent of T
TR	=	Rightmost dependent of T
NL	=	Leftmost dependent of N
L_i	=	Next plus i input token
X.LEX	=	Word form of X
X.POS	=	Part-of-speech of X
X.DEP	=	Dependency type of X

Figure 2: Parser state features

After initialization, the parser is guaranteed to terminate after at most $2n$ transitions, given an input string of length n (Nivre, 2003). Moreover, the parser always constructs a dependency graph that is acyclic and projective. This means that the dependency graph given at termination is well-formed if and only if it is connected (Nivre, 2003). Otherwise, it is a set of connected components, each of which is a well-formed dependency graph for a substring of the original input.

The transition system defined above is nondeterministic in itself, since several transitions can often be applied in a given configuration. To construct deterministic parsers based on this system, we use classifiers trained on treebank data in order to predict the next transition (and dependency type) given the current configuration of the parser. In this way, our approach can be seen as a form of history-based parsing (Black et al., 1992; Magerman, 1995). In the experiments reported here, we use memory-based learning to train our classifiers.

3 Memory-Based Learning

Memory-based learning and problem solving is based on two fundamental principles: learning is the simple storage of experiences in memory, and solving a new problem is achieved by reusing solutions from similar previously solved problems (Daelemans, 1999). It is inspired by the nearest neighbor approach in statistical pattern recognition and artificial intelligence (Fix and Hodges, 1952), as well as the analogical modeling approach in linguistics (Skousen, 1989; Skousen, 1992). In machine learning terms, it can be characterized as a lazy learning method, since it defers processing of input until needed and processes input by combining stored data (Aha, 1997).

Memory-based learning has been successfully applied to a number of problems in natural language processing, such as grapheme-to-phoneme conversion, part-of-speech tagging, prepositional-phrase attachment, and base noun phrase chunking (Daelemans et al., 2002). Previous work on memory-based learning for deterministic parsing includes Veenstra and Daelemans (2000) and Nivre et al. (2004).

For the experiments reported in this paper, we have used the software package TiMBL (Tilburg Memory Based Learner), which provides a variety of metrics, algorithms, and extra functions on top of the classical k nearest neighbor classification kernel, such as value distance metrics and distance weighted class voting (Daelemans et al., 2003).

The function we want to approximate is a mapping f from configurations to parser actions, where each action consists of a transition and (except for **Shift** and **Reduce**) a dependency type:

$$f : \mathit{Config} \rightarrow \{\mathbf{LA}, \mathbf{RA}, \mathbf{RE}, \mathbf{SH}\} \times (R \cup \{\mathbf{nil}\})$$

Here *Config* is the set of all configurations and R is the set of dependency types. In order to make the problem tractable, we approximate f with a function $\hat{f}$ whose domain is a finite space of parser *states*, which are abstractions over configurations. For this purpose we define a number of features that can be used to define different models of parser state.

Figure 2 illustrates the features that are used to define parser states in the present study. The two central elements in any configuration are the token on top of the stack (T) and the next input token

(N), the tokens which may be connected by a dependency arc in the next configuration. For these tokens, we consider both the word form (T.LEX, N.LEX) and the part-of-speech (T.POS, N.POS), as assigned by an automatic part-of-speech tagger in a preprocessing phase. Next, we consider a selection of dependencies that may be present in the current arc relation, namely those linking T to its head (TH) and its leftmost and rightmost dependent (TL, TR), and that linking N to its leftmost dependent (NL),[3] considering both the dependency type (arc label) and the part-of-speech of the head or dependent. Finally, we use a lookahead of three tokens, considering only their parts-of-speech.

We have experimented with two different state models, one that incorporates all the features depicted in Figure 2 (Model 1), and one that excludes the parts-of-speech of TH, TL, TR, NL (Model 2). Models similar to model 2 have been found to work well for datasets with a rich annotation of dependency types, such as the Swedish dependency treebank derived from Einarsson (1976), where the extra part-of-speech features are largely redundant (Nivre et al., 2004). Model 1 can be expected to work better for datasets with less informative dependency annotation, such as dependency trees extracted from the Penn Treebank, where the extra part-of-speech features may compensate for the lack of information in arc labels.

The learning algorithm used is the IB1 algorithm (Aha et al., 1991) with $k = 5$, i.e. classification based on 5 nearest neighbors.[4] Distances are measured using the modified value difference metric (MVDM) (Stanfill and Waltz, 1986; Cost and Salzberg, 1993) for instances with a frequency of at least 3 (and the simple overlap metric otherwise), and classification is based on distance weighted class voting with inverse distance weighting (Dudani, 1976). These settings are the result of extensive experiments partially reported in Nivre et al. (2004). For more information about the different parameters and settings, see Daelemans et al. (2003).

4 Experiments

The data set used for experimental evaluation is the standard data set from the Wall Street Journal section of the Penn Treebank, with sections 2–21 used for training and section 23 for testing (Collins, 1999; Charniak, 2000). The data has been converted to dependency trees using head rules (Magerman, 1995; Collins, 1996). We are grateful to Yamada and Matsumoto for letting us use their rule set, which is a slight modification of the rules used by Collins (1999). This permits us to make exact comparisons with the parser of Yamada and Matsumoto (2003), but also the parsers of Collins (1997) and Charniak (2000), which are evaluated on the same data set in Yamada and Matsumoto (2003).

One problem that we had to face is that the standard conversion of phrase structure trees to dependency trees gives unlabeled dependency trees, whereas our parser requires labeled trees. Since the annotation scheme of the Penn Treebank does not include dependency types, there is no straightforward way to derive such labels. We have therefore experimented with two different sets of labels, none of which corresponds to dependency types in a strict sense. The first set consists of the function tags for grammatical roles according to the Penn II annotation guidelines (Bies et al., 1995); we call this set G. The second set consists of the ordinary bracket labels (S, NP, VP, etc.), combined with function tags for grammatical roles, giving composite labels such as NP-SBJ; we call this set B. We assign labels to arcs by letting each (non-root) word that heads a phrase P in the original phrase structure have its incoming edge labeled with the label of P (modulo the set of labels used). In both sets, we also include a default label DEP for arcs that would not otherwise get a label. This gives a total of 7 labels in the G set and 50 labels in the B set. Figure 1 shows a converted dependency tree using the B labels; in the corresponding tree with G labels NP-SBJ would be replaced by SBJ, ADVP and VP by DEP.

We use the following metrics for evaluation:

1. **Unlabeled attachment score (UAS):** The proportion of words that are assigned the correct head (or no head if the word is a root) (Eisner, 1996; Collins et al., 1999).

2. **Labeled attachment score (LAS):** The proportion of words that are assigned the correct head and dependency type (or no head if the word is a root) (Nivre et al., 2004).

3. **Dependency accuracy (DA):** The proportion of non-root words that are assigned the correct head (Yamada and Matsumoto, 2003).

4. **Root accuracy (RA):** The proportion of root words that are analyzed as such (Yamada and Matsumoto, 2003).

[3]Given the parsing algorithm, N can never have a head or a right dependent in the current configuration.

[4]In TiMBL, the value of k in fact refers to k nearest distances rather than k nearest neighbors, which means that, even with $k = 1$, the nearest neighbor set can contain several instances that are equally distant to the test instance. This is different from the original IB1 algorithm, as described in Aha et al. (1991).

5. **Complete match (CM):** The proportion of sentences whose unlabeled dependency structure is completely correct (Yamada and Matsumoto, 2003).

All metrics except CM are calculated as mean scores per word, and punctuation tokens are consistently excluded.

Table 1 shows the attachment score, both unlabeled and labeled, for the two different state models with the two different label sets. First of all, we see that Model 1 gives better accuracy than Model 2 with the smaller label set G, which confirms our expectations that the added part-of-speech features are helpful when the dependency labels are less informative. Conversely, we see that Model 2 outperforms Model 1 with the larger label set B, which is consistent with the hypothesis that part-of-speech features become redundant as dependency labels get more informative. It is interesting to note that this effect holds even in the case where the dependency labels are mostly derived from phrase structure categories.

We can also see that the unlabeled attachment score improves, for both models, when the set of dependency labels is extended. On the other hand, the labeled attachment score drops, but it must be remembered that these scores are not really comparable, since the number of classes in the classification problem increases from 7 to 50 as we move from the G set to the B set. Therefore, we have also included the labeled attachment score restricted to the G set for the parser using the B set (BG), and we see then that the attachment score improves, especially for Model 2. (All differences are significant beyond the .01 level; McNemar's test.)

Table 2 shows the dependency accuracy, root accuracy and complete match scores for our best parser (Model 2 with label set B) in comparison with Collins (1997) (Model 3), Charniak (2000), and Yamada and Matsumoto (2003).[5] It is clear that, with respect to unlabeled accuracy, our parser does not quite reach state-of-the-art performance, even if we limit the competition to deterministic methods such as that of Yamada and Matsumoto (2003). We believe that there are mainly three reasons for this. First of all, the part-of-speech tagger used for preprocessing in our experiments has a lower accuracy than the one used by Yamada and Matsumoto (2003) (96.1% vs. 97.1%). Although this is not a very interesting explanation, it undoubtedly accounts for part of the difference. Secondly, since our parser makes crucial use of dependency type information in predicting the next action of the parser, it is very likely that it suffers from the lack of real dependency labels in the converted treebank. Indirect support for this assumption can be gained from previous experiments with Swedish data, where almost the same accuracy (85% unlabeled attachment score) has been achieved with a treebank which is much smaller but which contains proper dependency annotation (Nivre et al., 2004).

A third important factor is the relatively low root accuracy of our parser, which may reflect a weakness in the one-pass parsing strategy with respect to the global structure of complex sentences. It is noteworthy that our parser has lower root accuracy than dependency accuracy, whereas the inverse holds for all the other parsers. The problem becomes even more visible when we consider the dependency and root accuracy for sentences of different lengths, as shown in Table 3. Here we see that for really short sentences (up to 10 words) root accuracy is indeed higher than dependency accuracy, but while dependency accuracy degrades gracefully with sentence length, the root accuracy drops more drastically (which also very clearly affects the complete match score). This may be taken to suggest that some kind of preprocessing in the form of clausing may help to improve overall accuracy.

Turning finally to the assessment of labeled dependency accuracy, we are not aware of any strictly comparable results for the given data set, but Buchholz (2002) reports a labeled accuracy of 72.6% for the assignment of grammatical relations using a cascade of memory-based processors. This can be compared with a labeled attachment score of 84.4% for Model 2 with our B set, which is of about the same size as the set used by Buchholz, although the labels are not the same. In another study, Blaheta and Charniak (2000) report an F-measure of 98.9% for the assignment of Penn Treebank grammatical role labels (our G set) to phrases that were correctly parsed by the parser described in Charniak (2000). If null labels (corresponding to our DEP labels) are excluded, the F-score drops to 95.7%. The corresponding F-measures for our best parser (Model 2, BG) are 99.0% and 94.7%. For the larger B set, our best parser achieves an F-measure of 96.9% (DEP labels included), which can be compared with 97.0% for a similar (but larger) set of labels in Collins (1999).[6] Although none of the previous results on labeling accuracy is strictly comparable to ours, it nevertheless seems fair to conclude that the

[5]The information in the first three rows is taken directly from Yamada and Matsumoto (2003).

[6]This F-measure is based on the recall and precision figures reported in Figure 7.15 in Collins (1999).

	Model 1			Model 2		
	G	**B**	**BG**	**G**	**B**	**BG**
UAS	86.4	86.7		85.8	87.1	
LAS	85.3	84.0	85.5	84.6	84.4	86.0

Table 1: Parsing accuracy: Attachment score (BG = evaluation of B restricted to G labels)

	DA	RA	CM
Charniak	92.1	95.2	45.2
Collins	91.5	95.2	43.3
Yamada & Matsumoto	90.3	91.6	38.4
Nivre & Scholz	87.3	84.3	30.4

Table 2: Comparison with related work (Yamada and Matsumoto, 2003)

labeling accuracy of the present parser is close to the state of the art, even if its capacity to derive correct structures is not.

5 Conclusion

This paper has explored the application of a data-driven dependency parser to English text, using data from the Penn Treebank. The parser is deterministic and uses a linear-time parsing algorithm, guided by memory-based classifiers, to construct labeled dependency structures incrementally in one pass over the input. Given the difficulty of extracting labeled dependencies from a phrase structure treebank with limited functional annotation, the accuracy attained is fairly respectable. And although the structural accuracy falls short of the best available parsers, the labeling accuracy appears to be competitive.

The most important weakness is the limited accuracy in identifying the root node of a sentence, especially for longer sentences. We conjecture that an improvement in this area could lead to a boost in overall performance. Another important issue to investigate further is the influence of different kinds of arc labels, and in particular labels that are based on a proper dependency grammar. In the future, we therefore want to perform more experiments with genuine dependency treebanks like the Prague Dependency Treebank (Hajic, 1998) and the Danish Dependency Treebank (Kromann, 2003). We also want to apply dependency-based evaluation schemes such as the ones proposed by Lin (1998) and Carroll et al. (1998).

Acknowledgements

The work presented in this paper has been supported by a grant from the Swedish Research Council (621-2002-4207). The memory-based classifiers used in the experiments have been constructed using the Tilburg Memory-Based Learner (TiMBL) (Daelemans et al., 2003). The conversion of the Penn Treebank to dependency trees has been performed using head rules kindly provided by Hiroyasu Yamada and Yuji Matsumoto.

References

D. W. Aha, D. Kibler, and M. Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6:37–66.

D. Aha, editor. 1997. *Lazy Learning*. Kluwer.

A. Bies, M. Ferguson, K. Katz, and R. MacIntyre. 1995. Bracketing guidelines for Treebank II style, Penn Treebank project. University of Pennsylvania, Philadelphia.

E. Black, F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, and S. Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 5th DARPA Speech and Natural Language Workshop*.

D. Blaheta and E. Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of NAACL*, pages 234–240.

S. Buchholz. 2002. *Memory-Based Grammatical Relation Finding*. Ph.D. thesis, University of Tilburg.

J. Carroll, E. Briscoe, and A. Sanfilippo. 1998. Parser evaluation: A survey and a new proposal. In *Proceedings of LREC*, pages 447–454, Granada, Spain.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*.

M. Collins, J. Hajič, E. Brill, L. Ramshaw, and C. Tillmann. 1999. A Statistical Parser of Czech. In *Proceedings of ACL*, pages 505–512, University of Maryland, College Park, USA.

	DA	**RA**	**CM**
≤ 10	93.7	96.6	83.6
11–20	88.8	86.4	39.5
21–30	87.4	83.4	20.8
31–40	86.8	78.1	9.9
≥ 41	84.6	74.9	1.8

Table 3: Accuracy in relation to sentence length (number of words)

M. Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of ACL*, pages 184–191, Santa Cruz, CA.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL*, pages 16–23, Madrid, Spain.

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

S. Cost and S. Salzberg. 1993. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.

W. Daelemans, A. van den Bosch, and J. Zavrel. 2002. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34:11–43.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2003. Timbl: Tilburg memory based learner, version 5.0, reference guide. Technical Report ILK 03-10, Tilburg University, ILK.

W. Daelemans. 1999. Memory-based language processing. Introduction to the special issue. *Journal of Experimental and Theoretical Artificial Intelligence*, 11:287–292.

S. A. Dudani. 1976. The distance-weighted k-nearest neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6:325–327.

J. Einarsson. 1976. Talbankens skriftspråkskonkordans. Lund University.

J. M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*, Copenhagen, Denmark.

E. Fix and J. Hodges. 1952. Discriminatory analysis: Nonparametric discrimination: Consistency properties. Technical Report 11, USAF School of Aviation Medicine, Randolph Field, Texas.

J. Hajic. 1998. Building a syntactically annotated corpus: The prague dependency treebank. In *Issues of Valency and Meaning*, pages 106–132. Karolinum.

M. T. Kromann. 2003. The Danish dependency treebank and the DTAG treebank tool. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories*, pages 217–220, Växjö, Sweden.

T. Kudo and Y. Matsumoto. 2000. Japanese dependency structure analysis based on support vector machines. In *Proceedings of EMNLP/VLC*, Hongkong.

D. Lin. 1998. Dependency-based evaluation of MINIPAR. In *Proceedings of LREC*.

D. M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of ACL*, pages 276–283, Boston, MA.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

J. Nivre, J. Hall, and J. Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL*, pages 49–56.

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

K. Oflazer. 2003. Dependency parsing with an extended finite-state approach. *Computational Linguistics*, 29:515–544.

R. Skousen. 1989. *Analogical Modeling of Language*. Kluwer.

R. Skousen. 1992. *Analogy and Structure*. Kluwer.

C. Stanfill and D. Waltz. 1986. Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1228.

V. N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag.

J. Veenstra and W. Daelemans. 2000. A memory-based alternative for connectionist shift-reduce parsing. Technical Report ILK-0012, University of Tilburg.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206, Nancy, France.