Optimizing a Deterministic Dependency Parser for Unrestricted Swedish Text

Joakim Nivre
Växjö University
School of Mathematics and Systems Engineering
SE-35195 VÄXJÖ
`Joakim.Nivre@msi.vxu.se`

Abstract

This paper explores the use of machine learning in optimizing a syntactic parser for unrestricted Swedish text. The syntactic analysis is based on dependency grammar and the original parsing algorithm deterministically chooses the closest possible attachment for every word in the input string. The goal of the optimization is that the parser should learn, from a training set of correctly analyzed sentences, when to postpone the attachment. Experiments using a variant of k nearest neighbor learning show that a small but significant improvement in precision can be achieved in this way.

1 Introduction

Syntactic parsing of unrestricted natural language text is a crucial task in large-scale natural language processing systems such as machine translation systems. In addition, it has a great potential for improving the quality of information retrieval, information extraction and content management systems. Two major obstacles to high-quality parsing of unrestricted text have traditionally been the massive ambiguity inherent in natural language grammars and the lack of robustness in parsing systems using these grammars.

Deterministic dependency parsing has recently been proposed as a robust and efficient method for dealing with these problems and the results so far are promising. Thus, Nivre and Nilsson [24] report both precision and recall above 80% for parsing unrestricted Swedish text with very simple grammar rules, using a parsing algorithm that selects the closest possible attachment for every word in the dependency graph. In order to improve the accuracy further, a more flexible parsing strategy may be required, where the closest possible link is usually – but not always – chosen. In this paper, I will investigate the possibility of using machine learning techniques to determine when the parser should postpone the attachment.

The paper is structured in the following way. In section 2, I introduce the framework of deterministic dependency parsing proposed by Nivre and Nilsson [24] and analyze the typical errors resulting from the closest-first strategy. In section 3, I discuss the use of machine learning to improve parsing and in particular the choice of attributes to be used as input to the learning process. In

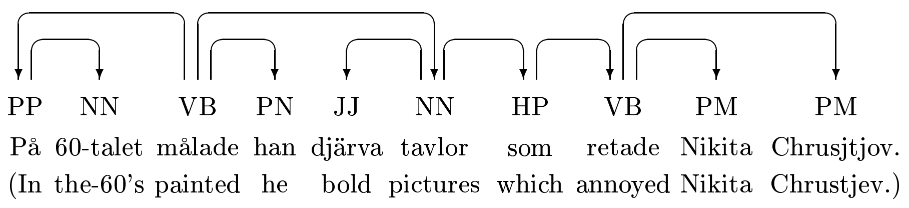

PP NN VB PN JJ NN HP VB PM PM
På 60-talet målade han djärva tavlor som retade Nikita Chrusjtjov.
(In the-60's painted he bold pictures which annoyed Nikita Chrustjev.)

Figure 1: Dependency graph for Swedish sentence

section 4, I present an experiment where a variant of k nearest neighbor learning is used to achieve a small but significant improvement of precision. In section 5, I conclude with some suggestions for further research.

2 Deterministic Dependency Parsing

Dependency parsing is based on the old linguistic tradition of dependency grammar, which comprises a large family of grammatical theories and formalisms that share certain basic assumptions about syntactic structure, in particular the assumption that syntactic structure consists of *lexical nodes* linked by binary relations called *dependencies* (see, e.g., Tesnière [30], Sgall et al. [26], Mel'čuk [22], Hudson [20]). Thus, the common formal property of dependency structures, as compared to the more common syntactic representations based on constituency (or phrase structure), is the lack of phrasal nodes.

In a dependency structure, every lexical node is dependent on at most one other lexical node, usually called its *head* or *regent*, which means that the structure can be represented as a directed graph, with nodes representing lexical elements and edges representing dependency relations. Normally we also require that the graph is connected and acyclic, which means that it will in fact be a rooted tree with the root node being the head of the sentence. Figure 1 shows a dependency graph for the Swedish sentence *På 60-talet målade han djärva tavlor som retade Nikita Chrusjtjov*, with the finite verb *målade* as the root node. The label occurring above each word is its lexical category, or part of speech; PP for preposition, NN for noun, VB for finite verb, etc.

Most formalizations of dependency grammar use rules that specify whole configurations of dependents for a given head, using some notion of *valence frames* (Hays [19], Gaifman [17], Carroll and Charniak [5], Sleator and Temperley [27, 28], Barbero et al. [1], Eisner [16], Debusmann [14]). Nivre and Nilsson [24] use a much weaker formalism, where only binary relations between heads and dependents can be specified, using rules of the following form:

$$D \leftarrow H$$
$$H \rightarrow D$$

These rules say that a word h of category H can be the syntactic head of a word d of category D if d *precedes* ($\leftarrow$) or *follows* ($\rightarrow$) h. Similar but not identical grammar formalisms have previously been proposed by Covington [12, 13] and Courtin and Genthial [11].

Nivre and Nilsson [24] consider three different parsing algorithms, which take as input a grammar G and and a string of words $w_1, \ldots, w_n$, tagged with parts-of-speech $t_1, \ldots, t_n$, and builds a dependency graph by adding edges (w_i, w_j) that are compatible with the rules of G. The algorithms are all deterministic in the sense that once an edge has been added to the dependency graph it can never be removed and will therefore block the addition of other possible edges, given the constraint that each word can have at most one head. Given that syntactic relations tend to be local, all three algorithms have a preference for closer links over more distant ones. But they differ in the way that this preference is balanced against other constraints.

When tested on a random sample of 142 sentences from the Stockholm-Umeå Corpus of written Swedish (SUC [29]), manually annotated with the correct dependency graph for each sentence, the best algorithm achieved a precision of 85.5% and a recall of 83.9% on previously unseen data (Nivre and Nilsson [24]). These results are comparable to the best results published for dependency parsing of unrestricted texts, (see e.g. Eisner [15], Collins et al. [10]), although there are no directly comparable results available for texts in Swedish.

The highest-scoring algorithm is the so-called *projective* algorithm, which has the following overall structure:

```
**for** width $k = 1$ to $n - 1$ **do**
    **for** position $i = 1$ to $n - k$ **do**
        LINK($w_i, w_{i+k}$)
```

The single parsing operation used is LINK(w_i, w_j), which tries to establish a dependency link between two words w_i and w_j. Using the notation $w : C$ to mean that the word w belongs to the category C, and $R \in G$ to mean that R is a rule of the current grammar G, this operation can be defined as follows:

```
**if** $w_i$ has no head
    and $w_i : D$ and $w_j : H$ and $D \leftarrow H \in G$
    and $w_i$ and $w_j$ are accessible
    and there is no path from $w_i$ to $w_j$
**then**
    add the edge $(w_j, w_i)$
**else if** $w_j$ has no head
    and $w_i : H$ and $w_j : D$ and $H \rightarrow D \in G$
    and $w_i$ and $w_j$ are accessible
    and there is no path from $w_j$ to $w_i$
**then**
    add the edge $(w_i, w_j)$
```

A node w_j is *accessible* iff there is no edge (w_i, w_k) such that $i < j < k$. The effect of the accessibility constraint, in conjunction with the condition that there be no path from dependent to head, guarantees that the resulting dependency graph is acyclic and free from crossing edges, a property which is known as *projectivity* in dependency grammar, hence the name of the algorithm.

The projective algorithm constructs a dependency graph by linking each word to its closest possible regent proceeding left-to-right through the input and observing the constraints on projectivity. More precisely, it runs through the input words from left to right $n - 1$ times (where n is the number of words in the input), considering possible links of length k during iteration k. The number of iterations is therefore $O(n^2)$ but since the worst-case complexity of the LINK operation is $O(n)$, the running time of the algorithm is $O(n^3)$.

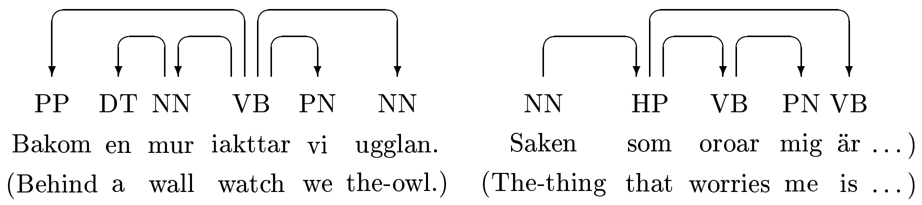
Precision and recall approaching 85% when parsing unrestricted text shows that the preference for close links is a sensible overall strategy. At the same time, it is a strategy that leads to error whenever the correct link is not the closest possible one. Some of the errors performed by the parser are well-known problems having to do with attachment ambiguity for prepositional phrases and other adjuncts, which are known to be hard for any natural language parser. Other errors seem to be more specific to the deterministic closest-first strategy, and the way it interacts with the very simple form of grammars used. Errors in the latter group fall mainly into three (partly overlapping) categories:

- **Valence violations:** Given the form of the grammar, there is no way to impose restrictions on the number of dependents of a single head. Thus, in the left hand example in Figure 2, the transitive verb *iakttar* is incorrectly linked to three nominal dependents, which is one too many. Furthermore, the incorrect link between *iakttar* and *mur* blocks the correct link from *mur* to its prepositional head *bakom*.

- **Linking across syntactic barriers:** Despite the fact that most dependencies are local to the syntactic clause, there is nothing that prevents the parser from adding links across clause boundaries. For example, in the right hand sentence in Figure 2, the subjunction *som* is correctly determined to be the head of the finite verb *oroar* in the subordinate clause but incorrectly analyzed as the head of the finite verb *är* in the main clause. Moreover, this incorrect link also blocks the correct link from the finite verb *är* to its subject *saken* because of the ban on cyclic graphs.

- **Errors caused by elliptical constructions:** Dependency grammar presupposes that all syntactic constructions have a head, an assumption that is barely satisfied if we limit ourselves to canonical syntactic structures. However, in elliptical constructions it is often the case that the expected syntactic head is omitted, which causes severe problems for the parser. Cases in point are clauses without a finite verb and noun phrases without a head noun, both of which are exemplified in the sentence *Hon läste den tjocka boken och han den tunna med stor stil* (She read the thick book and he the thin [one] with large print).

Coping with errors in the third category probably requires major changes either in the grammar or in the parsing algorithm (or both), and these problems may even be best handled in a separate post-processing stage. Errors in the first two categories, however, should be tractable by making the parser sensitive to the number of dependents of a given head, and to the presence of clause boundaries in the input string. In this paper, we will investigate the possibility of using machine learning to achieve this goal.



PP DT NN VB PN NN
Bakom en mur iakttar vi ugglan.
(Behind a wall watch we the-owl.)
NN HP VB PN VB
Saken som oroar mig är ...)
(The-thing that worries me is ...)

Figure 2: Two typical parsing errors

3 Machine Learning

The goal of machine learning is to develop computer systems that improve their performance with increased experience, i.e. systems that in some sense learn from their experience. In the last ten to fifteen years, there has been a dramatic increase in the use of machine learning techniques in different areas of natural language processing. As far as syntactic parsing is concerned, machine learning has mainly been used to discover rules for syntactic analysis, or to estimate probabilistic parameters for such rules (see, e.g., Pereira and Schabes [25], Brill [2], Charniak [6, 7], Collins [8, 9]), rather than to train syntactic parsers directly (see, however, Briscoe and Carroll [4] and Magerman [21]).

In order to have a well-defined machine learning problem, me must specify a task T to be performed, a measure P to assess the performance of the system at T, and a type of learning experience E to be used to improve the performance of the system (Mitchell [23]). In the current setting, T is the task of parsing unrestricted Swedish text and P is the precision and recall achieved at this task (measured against an empirical gold standard obtained by manual annotation). The learning experience E will be the parser's own performance on Swedish text, compared to a manually annotated gold standard of the same kind that is used in evaluation.

In order to have a complete learning system, we must also define the exact type of knowledge to be learned, a representation of this target knowledge, and a learning mechanism. We will postpone discussion of the learning mechanism and concentrate first on the knowledge to be learned. This is often conceptualized as a target function to be approximated, i.e. some function $f : I \to O$, where I is the space of possible inputs and O is the space of possible outputs, and where the learnt approximation $\hat{f} : I \to O$ can be used to improve the system's performance at the relevant task (cf. Mitchell [23], Hastie et al. [18]).

In the present context, we want the parser to improve its performance by learning when a possible edge should be added to the dependency graph and when it should be omitted in favor of a more distant link. Thus, it seems reasonable to take our target function f to be a binary decision function, which takes value 1 if a given edge should be added and value 0 if it should not be added. The input to this function will be the state of the parser at decision time. Thus, we can say that $f : Q \to \{0, 1\}$, where Q is the set of all possible parser states.

In order to choose a representation of this function we must first decide what aspects of parser states should be taken into account. Based on the analysis of typical parser errors in the preceding section, we hypothesize that the following properties of the parser state could be relevant for the decision of adding an edge (w_i, w_j) or not:

1. The grammar rule licensing the edge, consisting of a head category H, a dependent category D, and a direction d (where $w_i : H$, $w_j : D$ and d is either $\leftarrow$ or $\rightarrow$).

2. The number of dependents of w_i at decision time, possibly separated into left and right dependents.

3. The presence of any clause boundaries or other syntactic barriers between w_i and w_j.

The presence of other dependents of the potential head w_i is relevant to avoid valence violations, although the parser only has access to the dependents that exist at decision time, which may not be the full set of dependents in the final dependency graph. Similarly, syntactic barriers can only be detected with good accuracy after the parsing is complete, which means that we have to rely on indirect evidence from the presence of certain syntactic categories. Subjunctions (SN), as well as interrogative and relative pronouns (HP) and adverbs (HA), are fairly good indicators of clause boundaries. In addition, finite verbs (VB) may be considered as clues in this context. Finally, the occurrence of words belonging to the same category as w_i or w_j may be relevant as barriers in certain cases.

Based on these considerations, the following set of attributes were selected to represent parser states:

Attribute	**Meaning**
LCat	Category of left word w_l
RCat	Category of right word w_r
Dir	Direction of dependency ($\leftarrow$ or $\rightarrow$)
Dist	Distance between w_l and w_r (Dist $= r - l$)
LDep	Number of left dependents of potential head w_h
RDep	Number of right dependents of potential head w_h
CB	Number of HA, HP and SN between w_l and w_r
VB	Number of VB between w_l and w_r
LBar	Number of LCat between w_l and w_r
RBar	Number of RCat between w_l and w_r

This means that training instances in the learning process will be represented as pairs (X, Y), where X is a list of values for the attributes listed above, and Y is 1 or 0 depending on whether the relevant edge is part of the correct dependency graph or not.

For the experiment reported in the next section, a version of k-nearest neighbor learning was used to approximate the target function f. Nearest neighbor learning is a simple yet effective learning method, which has been applied successfully to a variety of natural language processing problems. It therefore seemed like a natural choice when starting to explore the potential of machine learning for parser optimization.

Given a set of training instances $T = (\langle x_1, y_1\rangle, \ldots, \langle x_n, y_n\rangle)$, the k-nearest neighbor approximation is defined as follows (Mitchell [23], Hastie [18]):

$$k\mathrm{NN}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

where $N_k(x)$ is the neighborhood defined by the k closest points x_i in the training sample, according to a suitable distance measure. In order to turn this into a discrete decision function, we use the following mapping:

$$\hat{f}(x) = \begin{cases} 1 & \text{if } k\text{NN}(x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

The distance measure used is the Euclidean distance between the points in n-dimensional space defined by the numerical attributes of each instance:

$$d(v_i, v_j) = \sqrt{\sum_{r=1}^{n} (a_r(v_i) - a_r(v_j))^2}$$

where v_i is the n-dimensional vector defined by the numerical attributes of x_i and $a_r(v)$ denotes the value of the rth attribute of vector v. In our case, the numerical attributes are Dist, LDep, RDep, CB, FV, LBar, RBar, so $n = 7$.

4 Experiment

In order to test the viability of the machine learning approach to parser optimization, an experiment was performed using data from the Stockholm-Umeå Corpus of written Swedish (SUC [29]). SUC is annotated for parts of speech (and manually corrected) and can therefore be used as direct input to the parser. (Otherwise a part-of-speech tagger must be used to preprocess the input.)

Two independent data sets were sampled, each consisting of roughly 2000 words, corresponding to 115 and 142 sentences, respectively. Both samples were manually annotated with dependency graphs by the author. The first data set was used as training data, while the second data set was used for evaluation. (Apart from the fact that a few annotation errors have been corrected, the two data sets are identical to the ones used in Nivre and Nilsson [24].)

When annotating the sentences, major delimiters such as colon and semicolon were treated as barriers for dependency relations. This means that the strings occurring on each side of such a delimiter were treated as separate parse units even if they were not strictly speaking separate sentences. Moreover, text occurring in parentheses was treated as invisible to the surrounding text, in the sense that dependency relations were permitted across and within – but not into or out of – the parenthesized text.

It is also worth mentioning some of the principles used in choosing between alternative structural analyses for the annotation:

- Syntactic dependencies are preferred over semantic ones, meaning among other things that:

    1. Nouns depend on prepositions.
    2. Finite verbs depend on subjunctions and fronted *Wh*-words.
    3. Main verbs depend on auxiliary verbs.

- Coordinated items are treated as multiple dependents of their mutual head (if any), while the coordinating conjunctions are left unattached.

- Multi-word proper names are treated as coordinated items.
- Nominal appositive constructions are consistently analyzed as left-headed.

The grammar used in the experiments was the grammar that achieved the highest accuracy together with the projective parsing algorithm in Nivre and Nilsson [24]. This grammar was constructed by iteratively removing low precision rules from an initial hand-crafted grammar G_0 until no further improvement was possible with respect to the training data set. The initial grammar G_0 contained a total of 139 rules, divided into 100 left-headed rules (of the form $H \rightarrow D$) and 39 right-headed rules (of the form $D \leftarrow H$). The optimal grammar G_{12} was obtained after removing 12 rules, 9 left-headed and 3 right-headed, yielding a total of 127 rules in the grammar.

The performance measures used were *precision* and *recall*, calculated per sentence by comparing the dependency graphs built by the parser to the manually annotated gold standard:

$$\begin{aligned} \text{Precision} &= \frac{\mid \text{Correct edges in parse} \mid}{\mid \text{Edges in parse} \mid} \\ \text{Recall} &= \frac{\mid \text{Correct edges in parse} \mid}{\mid \text{Edges in gold standard} \mid} \end{aligned}$$

The overall precision and recall were then calculated as the mean precision and recall over all sentences.

Training instances for the learning algorithm were created by running the projective parsing algorithm with grammar G_{12} on the training data sample from SUC. For each edge added by the parser, the ten input attributes were computed and the output value was set to 1 if the edge was correct according to the manually annotated gold standard and to 0 otherwise. In this way a training data set of 1461 instances was created.

The projective parsing algorithm was modified so that an edge is added to the dependency graph only if $\hat{f}(x) = 1$, where x is the current parser state, as represented by the ten attribute values, and $\hat{f}$ is the k nearest neighbor approximation defined in the preceding section, restricted to the set of instances that agree with x in their values for LCat, RCat and Dir, i.e. parser states where the same grammar rule is considered. This parser is called kNN in the following. Different values for k were tested, but the optimal value turned out to be $k = 2$ and this value was used throughout.

In the kNN parser, the presence of syntactic barriers, represented by positive values for the attribute CB (and possibly VB), will only influence the parser's decision if the k nearest neighbors involving the same grammar rule also has these barriers present. However, in many cases these barriers – unlike valence constraints – are not limited to particular rules and categories but apply across the board. Therefore, a second parser was constructed, where a nonzero value for the CB attribute always blocks the addition of an edge, irrespective of the k nearest neighbor value. In other words, this parser, called kNN*c*, uses the following function approximation:

$$\hat{f}(x) = \begin{cases} 0 & \text{if CB} > 1 \text{ or } k\text{NN}(x) < 0.5 \\ 1 & \text{otherwise} \end{cases}$$

By letting a nonzero value for either the CB or the VB attribute act as a global barrier, we obtained a third parser kNN*cv*. Finally, the original parser with

Parser	**Precision**	**Recall**
Baseline	85.3	82.9
kNN	85.9	83.1
$k$NN$c$	86.1	83.3
$k$NN$cv$	86.7	82.9

Table 1: Precision and recall for four different parsers

the unmodified projective algorithm was included as a baseline. (Note that the results obtained for the baseline parser differ by a small fraction from the ones reported in Nivre and Nilsson [24]. This is due to the correction of a few errors in the manually annotated gold standard.)

The precision and recall of the four parsers compared can be seen in Table 1. All three kNN parsers have a higher precision score than the baseline parser, although the difference is statistically significant only for $k$NN$c$ and $k$NN$cv$ (paired t-test, $\alpha = 0.05$). For recall there are only minor differences between the four parsers, none of which is significant.

The improvement in precision from 85.3% (baseline) to 86.7% ($k$NN$cv$) represents a 10% error reduction but is still not very impressive, and the interesting question is why there is not a bigger improvement. One possible answer lies in the sparseness of the training data. With a training set of 1461 instances and a grammar containing 127 rules there are on average only about ten instances per rule. Moreover, some of these rules occur very rarely, which means that the training data for these rules will be sparse, with the effect that the nearest neighbors may in fact be quite distant. The fact that $k = 2$ gave the best results may be an indication that this is at least part of the answer, since a larger value for k will bring even more distant neighbors into the picture.

A second possible explanation is the choice of attributes for representing the target function. It is possible that another selection of features would lead to a better approximation and a larger improvement of parsing accuracy. A third possibility is the choice of learning mechanism, given that nearest neighbor methods can be very sensitive to sparse data. (In the terminology of machine learning they have low bias but high variance.) A fourth alternative is that we are already close to the ceiling in terms of the parsing accuracy that can be achieved with the simple projective algorithm and the very weak grammar rules used so far. In this case, we may have to change either the form of the grammar or the parsing algorithm, or both, in order to improve accuracy further.

5 Conclusion

In principle, the experiment reported in the previous section shows that it is possible to use machine learning to improve the performance of a deterministic dependency parser. In practice, this is of little interest as long as the improvement is only marginal. More research is needed to determine whether this approach is only of theoretical interest or whether it can lead to substantially better parsing quality.

First of all, we need to increase the amount of training data available to the learning algorithm in order to reduce the variance in the function approximation. Regardless of what learning method we ultimately decide to use, the generalization performance is bound to improve with more training data.

Secondly, we may need to modify the representation of the target function, in particular the attributes representing the parser states. For example, we may consider not only dependents of the potential head but also potential heads of the dependent. That is, the decision to refrain from adding an edge (w_i, w_j) to the dependency graph is likely to be influenced by the availability of alternative regents for w_j.

Thirdly, it may be worthwhile to experiment with different learning methods, such as decision tree learning or Bayesian learning. Moreover, the choice of learning method is not independent of the target function representation, since learning methods differ in their sensitivity to the data sparseness that necessarily comes with using a more complex representation of the target function.

Fourthly, we may have to make modifications to the grammar formalism or even to the parsing algorithm itself. For instance, grammar rules could be made context-sensitive by adding constraints on what syntactic categories may or may not occur between the head and the dependent, or we might add constraints on the number of dependents for a given head category as a crude way of capturing valence constraints.

Finally, we may consider the possibility of using several passes over the input, where later passes can detect and correct errors introduced in previous passes, using methods such as transformation-based learning (Brill [3]).

Hopefully, carrying out this research program will lead to a substantial improvement in the quality of syntactic parsing for unrestricted natural language texts, to the point where the full potential of this technique can be exploited for the automated management of textual information, as suggested in the introduction.

References

[1] Cristina Barbero, Leonardo Lesmo, Vincenzo Lombardo, and Paola Merlo. Integration of syntactic and lexical information in a hierarchical dependency grammar. In Sylvain Kahane and Alain Polguère, editors, *Proceedings of the Workshop on Processing of Dependency-Based Grammars*, pages 58–67, Université de Montréal, Quebec, Canada, August 1998.

[2] Eric Brill. Transformation-based error-driven parsing. In *Proceedings Third International Workshop on Parsing Technologies*, 1993.

[3] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21:543–566, 1995.

[4] Ted Briscoe and John Carroll. Generalised probabilistic lr parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19:25–59, 1993.

[5] Glenn Carroll and Eugene Charniak. Two experiments on learning probabilistic dependency grammars from corpora. Technical Report TR-92, Department of Computer Science, Brown University, 1992.

[6] Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*. MIT Press, 1997.

[7] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings NAACL-2000*, 2000.

[8] Michael Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annatual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain, 1997.

[9] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[10] Michael Collins, Jan Hajič, Eric Brill, Lance Ramshaw, and Christoph Tillmann. A Statistical Parser of Czech. In *Proceedings of 37th ACL Conference*, pages 505–512, University of Maryland, College Park, USA, 1999.

[11] Jacques Courtin and Damien Genthial. Parsing with dependency relations and robust parsing. In Sylvain Kahane and Alain Polguère, editors, *Proceedings of the Workshop on Processing of Dependency-Based Grammars*, pages 95–101, Université de Montréal, Quebec, Canada, August 1998.

[12] Michael A. Covington. A dependency parser for variable-word-order languages. Technical Report AI-1990-01, University of Georgia, Athens, GA, 1990.

[13] Michael A. Covington. Discontinuous dependency parsing of free and fixed word order: Work in progress. Technical Report AI-1994-02, University of Georgia, Athens, GA, 1994.

[14] Ralph Debusmann. A declarative grammar formalism for dependency grammar. Master's thesis, Computational Linguistics, Universität des Saarlandes, November 2001.

[15] Jason M. Eisner. An empirical comparison of probability models for dependency grammar. Technical Report IRCS-96-11, Institute for Research in Cognitive Science, University of Pennsylvania, 1996.

[16] Jason M. Eisner. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*. Kluwer, 2000.

[17] Haim Gaifman. Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337, 1965.

[18] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[19] David G. Hays. Dependency theory: A formalism and some observations. *Language*, 40:511–525, 1964.

[20] Richard A. Hudson. *English Word Grammar*. Blackwell, 1990.

[21] David M. Magerman. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Boston, MA, 1995.

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

[23] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[24] Joakim Nivre and Jens Nilsson. Three algorithms for deterministic dependency parsing. Submitted to NODALIDA-2003, 2003.

[25] Fernando. C. Pereira and Yves Schabes. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, 1992.

[26] Petr Sgall, Eva Hajicova, and Jarmila Panevova. *The Meaning of the Sentence in Its Pragmatic Aspects.* Reidel, 1986.

[27] Daniel Sleator and Davy Temperley. Parsing English with a link grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University, Computer Science, 1991.

[28] Daniel Sleator and Davy Temperley. Parsing English with a link grammar. In *Third International Workshop on Parsing Technologies*, 1993.

[29] Stockholm Umeå Corpus. Version 1.0. Produced by Department of Linguistics, Umeå University and Department of Linguistics, Stockholm University. ISBN 91-7191-348-3., August 1997.

[30] Lucien Tesnière. *Éléments de syntaxe structurale.* Editions Klincksieck, 1959.