Aspects of Pattern-matching in Data-Oriented Parsing

Guy De Pauw
CNTS
University of Antwerp

Abstract

Data-Oriented Parsing (DOP) ranks among the best parsing schemes, pairing state-of-the art parsing accuracy to the psycholinguistic insight that larger chunks of syntactic structures are relevant grammatical and probabilistic units. Parsing with the DOP-model, however, seems to involve a lot of CPU cycles and a considerable amount of double work, brought on by the concept of multiple derivations, which is necessary for probabilistic processing, but which is not convincingly related to a proper linguistic backbone. It is however possible to reinterpret the DOP-model as a pattern-matching model, which tries to maximize the size of the substructures that construct the parse, rather than the probability of the parse. By emphasizing this memory-based aspect of the DOP-model, it is possible to do away with multiple derivations, opening up possibilities for efficient Viterbi-style optimizations, while still retaining acceptable parsing accuracy through enhanced context-sensitivity.

1 Introduction

The machine learning paradigm of Memory-Based Learning, based on the assumption that new problems are solved by direct reference to stored experiences of previously solved problems, has been successfully applied to a number of linguistic phenomena, such as part-of-speech tagging, NP-chunking and stress acquisition (consult Daelemans (1999) for an overview). To solve these particular problems, linguistic information needed to trigger the correct disambiguation, is encoded in a linear *feature value* representation and presented to a memory based learner, such as TiMBL (Daelemans et al., 1999).

Yet, many of the intricacies of the domain of syntax do not translate well to a linear representation, so that established MBL-methods are necessarily limited to low-level syntactic analysis, like the aforementioned NP-chunking task.

Data Oriented Parsing (Bod, 1999), a state-of-the art natural language parsing system, translates very well to a Memory Based Learning context. This paper describes a reinterpretation of the DOP-model, in which the *pattern-matching aspects* of the model are exploited, so that parses are analyzed by trying to match a new analysis to the largest possible substructures recorded in memory.

A short introduction to Data Oriented Parsing will be presented in Section 2, followed by an explanation of the term *pattern-matching* in the context of this paper. Section 4 describes the experimental setup and the corpus. The parsing phase that precedes the disambiguation phase will be outlined in Section 5 and a description of the 3 disambiguating models, PCFG, PMPG and the combined system PCFG+PMPG can be found in Sections 6, 7 and 8.

2 Data Oriented Parsing

Data Oriented Parsing, originally conceived by Remko Scha (Scha, 1990), has been successfully applied to syntactic natural language parsing by Rens Bod (1995), (1999). The aim of Data Oriented Parsing (henceforth DOP) is to develop a *performance model* of natural language, that models language use rather than some type of competence. It adapts the psycholinguistic insight that language users analyze sentences using previously registered constructions and that not only rewrite rules, but complete substructures of any given depth can be linguistically relevant units for parsing.

2.1 Architecture

The core of a DOP-system is its TREEBANK: an annotated corpus is used to induce all substructures of arbitrary depth, together with their respective probabilities, which is a expressed by

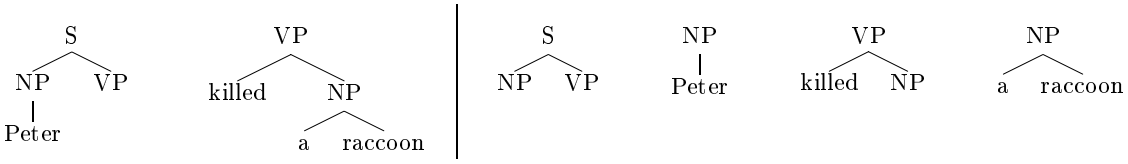


Figure 1: Multiple Derivations

its frequency in the TREEBANK relative to the number of substructures with the same root-node.

Figure 1 shows the combination operation that is needed to form the correct parse tree for the sentence *Peter killed a raccoon*. Given a treebank of substructures, the system tries to match the leftmost open node of a substructure that is consistent with the parse tree, with the top-node of another substructure, consistent with the parse tree.

Usually, different combinations of substructures are possible, as is indicated in Figure 1: in the example at the left-hand side the tree-structure can be built by combining an S-structure with a specified NP and a fully specified VP-structure. The right example shows another possible combination, where a parse tree is built by combining the minimal substructures. Note that these are consistent with ordinary rewrite-rules, such as S $\rightarrow$ NP VP.

One particular parse tree may thus consist of several different *derivations*. To find the probability of a derivation, we multiply the probabilities of the substructures that were used to form the derivation. To find the probability of a parse, we must in principle sum the probabilities of all its derivations.

It is computationally hardly tractable to consider all derivations for each parse. Since VITERBI optimization only succeeds in finding the most probable derivation as opposed to the most probable parse, the MONTE CARLO algorithm is introduced as a proper approximation that randomly generates a large number of derivations. The most probable parse is considered to be the parse that is most often observed in this derivation forest.

2.2 Experimental Results of DOP

The basic DOP-model, DOP1, was tested on a manually edited version of the ATIS-corpus (Marcus, Santorini, and Marcinkiewicz, 1993). The system was trained on 603 sentences (part-of-speech tag sequences) and evaluated on a test set of 75 sentences. Parse accuracy was used as an evaluation metric, expressing the percentage of sentences in the test set for which the parse proposed by the system is completely identical to the one in the original corpus. Different experiments were conducted in which maximum substructure size was varied. With DOP1-limited to a substructure-size of 1 (equivalent to a PCFG), parse accuracy is 47%. In the optimal DOP-model, in which substructure-size is not limited, a parse accuracy of **85%** is obtained.

2.3 Short Assessment of DOP

DOP1 in its optimal form achieves a very high parse accuarcy. The computational costs of the system, however, are equally high. Bod (1995) reported an average parse time of 3.5 hours per sentence. Even though current parse time is reported to be more reasonable, the optimal DOP algorithm in which no constrains are made on the size of substructures, may not yet be tractable for life-size corpora.

In a context-free grammar framework (consistent with DOP limited to a substructure-size of 1), there is only one way a parse tree can be formed (for example, the right hand side of Figure 1), meaning that there is only one derivation for a given parse tree. This allows efficient VITERBI style optimization.

To encode context-sensitivity in the system, DOP is forced to introduce multiple derivations, so that repeatedly the same parse tree needs to be generated, bringing about a lot of computational overhead.

Even though the use of larger syntactic contexts is highly relevant from a psycholinguistic point-of-view, there is no explicit preference being made for larger substructures in the DOP model. While the MONTE CARLO optimization scheme maximizes the probability of the derivations and seems to prefer derivations made up of larger substructures, it may be interesting to

Disambiguator	Parse Accuracy (/562)	%	F	Parse Accuracy on parsable sentences (/456)	%
PCFG	373	**66.4**	83.0	373	81.8
PMPG	327	**58.2**	75.1	327	71.7
PCFG+PMPG	402	**71.5**	85.2	402	88.2

Table 1: Experimental Results

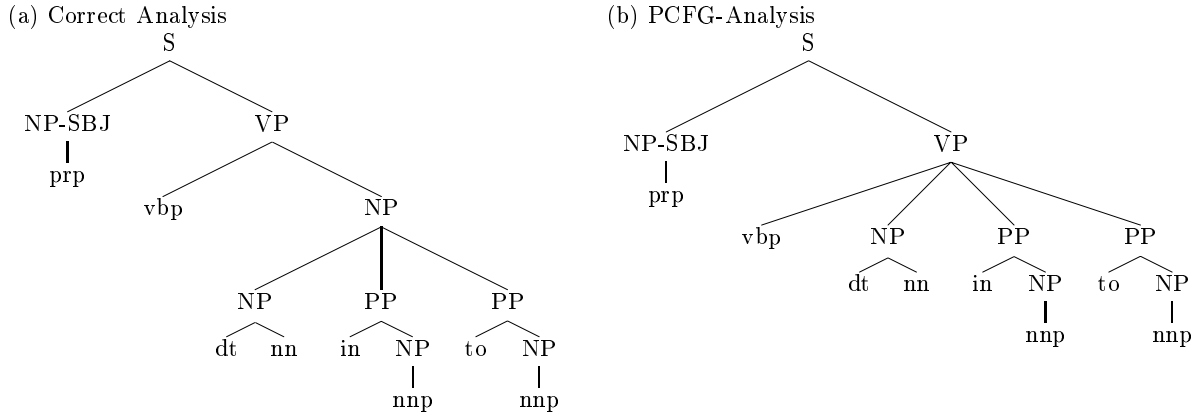


Figure 2: PCFG Error Analysis

see if we can make this assumption explicit.

3 Pattern-matching

When we look at natural language parsing from a memory-based point of view, one might say that a sentence is analyzed by looking up the most similar structure for the different analyses of that sentence in memory. The parsing system described in this paper tries to mimic this behavior by interpreting the DOP-model as a memory-based model, in which analyses are being matched with syntactic *patterns* recorded in memory. Similarity between the proposed analysis and the patterns in memory is computed according to:

- the number of patterns needed to construct a tree (to be minimized)
- the size of the patterns that are used to construct a tree (to be maximized)

The *nearest neighbor* for a given analysis can be defined as the derivation that shares the largest amount of common nodes.

4 The experimental Setup

10-fold cross-validation was used to appropriately evaluate the algorithms, as the dataset (see Section 4.1) is rather small. Like DOP1 the system is trained and tested on part-of-speech tag sequences. In a first phase, a simple bottom-up chart parser, trained on the training partitions, was used to generate parse forests for the part-of-speech tag sequences of the test partition. Next, the parse forests were sent to the 3 algorithms (henceforth the *disambiguators*) to order these parse forests, the first parse of the ordered parse forest being the one proposed by the disambiguator.

In this paper, 3 disambiguators are described:

- PCFG: simple Probabilistic Context-Free Grammar
- PMPG: the DOP approximation, Pattern-Matching Probabilistic Grammar
- PCFG+PMPG: a combined system, integrating PCFG and PMPG

The evaluation metric used is parse accuracy, but also the typical parser evaluation metric *F-measure* (precision/recall) is given as a means of reference to other systems.

4.1 The Corpus

The experiments were conducted on an edited version of the ATIS-II-corpus (Marcus, Santorini, and Marcinkiewicz, 1993), which consists of 578 sentences. Quite a lot of errors and inconsistencies were found, but not corrected, since we want our (probabilistic) system to be

able to deal with this kind of noise. Semantically oriented flags like -TMP and -DIR, most often used in conjunction with PP, have been removed, since there is no way of retrieving this kind of semantic information from the part-of-speech tags of the ATIS-corpus. Syntactic flags like -SBJ, on the other hand, have been maintained. Internal relations (denoted by numeric flags) were removed and for practical reasons, sentence-length was limited to 15 words max. The edited corpus retained 562 sentences.

5 Parsing

As a first phase, a bottom-up chart parser parsed the test set. This proved to be quite problematic, since overall, 106 out of 562 sentences (19%) could not be parsed, due to the sparseness of the grammar, meaning that the appropriate rewrite rule needed to construct the correct parse tree for a sentence in the test set, wasn't featured in the induced grammar. NP-annotation seemed to be the main cause for unparsability. An NP like *restriction code AP/57* is represented by the rewrite rule:

NP → NN NN sym sym sym CD CD

Highly specific and flat structures like these are scarce and are usually not induced from the training set when needed to parse the test set.

On-going research tries to implement grammatical smoothing as a solution to this problem, but one might also consider generating parse forests with an independent grammar, induced from the entire corpus (training set+testset) or a different corpus. In both cases, however, we would need to apply probabilistic smoothing to be able to assign probabilities to unknown structures/rules. Neither grammatical, nor probabilistic smoothing was implemented in the context of the experiments, described in this paper.

The sparseness of the grammar proves to be a serious bottleneck for parse accuracy, limiting our disambiguators to a maximum parse accuracy of 81%.

6 PCFG-experiments

A PCFG constructs parse trees by using simple rewrite-rules. The probability of a parse tree can be computed by multiplying the probabilities of the rewrite-rules that were used to construct the parse. Note that a PCFG is identical to DOP1 when we limit the maximum substructures size to 1, only allowing derivations of the type found at the right-hand side of Figure 1.

6.1 Experimental Results

The first line of Table 1 shows the results for the PCFG-experiments: 66.4% parse accuracy is an adequate result for this baseline model. We also look at parse accuracy for parsable sentences (an estimate of the parse accuracy we might get if we had a more suited parse forest generator) and we notice that we are able to achieve a 81.8% parse accuracy. This is already quite high, but on examining the parsed data, serious and fundamental limitations to the PCFG-model can be observed

6.2 Error Analysis

Figure 2, displays the most common type of mistake made by PCFG's. The correct parse tree could represent an analysis for the sentence:

I want a flight from Brussels to Toronto.

This example shows that a PCFG has a tendency to prefer flatter structures over embedded structures. This is a trivial effect of the mathematical formula used to compute the probability of a parse-tree: embedded structure require more rewrite rules, adding more factors to the multiplication, which will almost inevitably result in a lower probability.

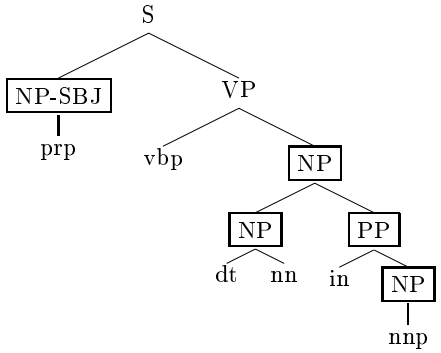
It is an unfortunate property of PCFG's that the number of nodes in the parse tree is inversely proportionate to its probability. One might be inclined to normalize a parse tree's probability relative to the number of nodes in the tree, but a more linguistically sound alternative is at hand: the enhancement of context sensitivity through the use of larger syntactic context within parse trees can make our disambiguator more robust.

7 PMPG-experiments

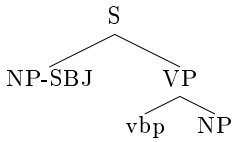
The Pattern-Matching Probabilistic Grammar is a memory-based interpretation of a DOP-model, in which a sentence is analyzed by matching the largest possible chunks of syntactic structure on the sentence. To compile parse trees into patterns, all substructures in the training set are encoded by assigning them specific indexes, NP@345 e.g. denoting a fully specified NP-structure. This approach was inspired by Goodman (1996), in which Goodman

unsuccessfully uses a system of indexed parse trees to transform DOP into an equivalent PCFG. The system of indexing (which is detailed in De Pauw (2000)) used in the experiments described in this paper, is however specifically geared towards encoding contextual information in parse trees.

Given an indexed training set, indexes can then be matched on a test set parse tree in a bottom-up fashion. In the following example, boxed nodes indicate nodes that have been retrieved from memory.



In this example we can see that an NP, consisting of a fully specified embedded NP and PP, has been completely retrieved from memory, meaning that the NP in its entirety can be observed in the training set. However, no VP was found that consists of a VBP and that particular NP. Disambiguating with PMPG consequently involves pruning all nodes retrieved from memory:



Finally, the probability for this pruned parse tree is computed in a PCFG-type manner, not adding the retrieved nodes to the product:

$$\text{P(parse)} = \text{P(S} \rightarrow \text{NP-SBJ VP)} \ . \ \text{P(VP} \rightarrow \text{vbp NP)}$$

7.1 Experimental Results

The results for the PMPG-experiments can be found on the second line of Table 1. On some partitions, PMPG performed insignificantly better than PCFG, but Table 1 shows that the results for the context sensitive scheme are much worse. 58.2% overall parse accuracy and 71.7% parse accuracy on parsable sentences indicates that PMPG is not a valid approximation of DOP's context-sensitivity.

7.2 Error Analysis

The dramatic drop in parsing accuracy calls for an error analysis of the parsed data. Figure 3 is a prototypical mistake PMPG has made. The correct analysis could represent a parse tree for a sentence like:

What flights can I get from Brussels to Toronto.

The PMPG analysis would never have been considered a likely candidate by a common PCFG. This particular sentence in fact was effortlessly disambiguated by the PCFG . Yet the fact that large chunks of tree-structure are retrieved from memory, make it the preferred parse for the PMPG. We notice for instance that a large part of the sentence can be matched on an SBAR structure, which has no relevance whatsoever.

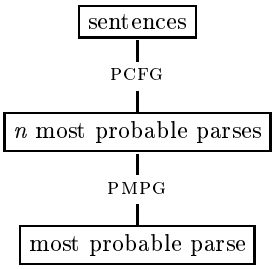
Clearly, PMPG overestimates substructure size as a feature for disambiguation. It's interesting however to see that it is a working implementation of context sensitivity, eagerly matching patterns from memory. At the same time, it has lost track of common-sense PCFG tactics. It is in the combination of the two that one may find a decent disambiguator and accurate implementation of context-sensitivity.

8 A Combined System (PMPG+PCFG)

Table 1 showed that 81.8% of the time, a PCFG finds the correct parse (for parsable sentences), meaning that the correct parse is at the first place in the ordered parse forest. **99%** of the time, the correct parse can be found among the 10 most probable parses in the ordered parse forest. This opens up a myriad of possibilities for optimization. One might for instance use a *best-first* strategy to generate only the 10 best parses, significantly reducing parse and disambiguation time. An optimized disambiguator might therefore include a preparatory phase in which a common-sense PCFG retains the most probable parses, so that a more sophisticated follow-up scheme need not bother with senseless analyses.

In our experiments, we combined the common-sense logic of a PCFG and used its output as the PMPG's input. This is a well-established technique usually referred to as *system combination* (see van Halteren, Zavrel, and Daelemans (1998) for an application of this

technique to part-of-speech tagging):



We are also presented with the possibility to assign a weight to each algorithm's decision. The probability of a parse can the be described with the following formula:

$$P(parse) = \frac{\prod_i P(\text{rewrite-rule})_i}{(\#\text{ non-indexed nodes})^n}$$

The weight of each algorithm's decision, as well as the number of most probable parses that are extrapolated for the pattern-matching algorithm, are parameters to be optimized. Future work will include evaluation on a validation set to retrieve the optimal values for these parameters.

8.1 Results

The third line in Table 1 shows that the combined system performs better than either one, with a parse accuracy of 71.5% and close to 90% parse accuracy on parsable sentences, which we can consider an approximation of results reported for DOP1. Error analysis shows that the combined system is indeed able to overcome difficulties of both algorithms. The example in Figure 2 as well as the example in Figure 3 were disambiguated correctly using the combined system

9 Future Research

Even though the PMPG shows a lot of promise in its parse accuracy, the following extensions need to be researched:

- Optimizing PMPG+PCFG for computational efficiency: the graph in Section 8 shows a possible optimized parsing system, in which a pre-processing PCFG generates the n most likely candidates to be extrapolated for the actual disambiguator. Full parse forests were generated for the experiments described in this paper, so that the efficiency gain of such a system cannot be properly estimated.
- PMPG+PCFG as an approximation needs to be compared to actual DOP, by having DOP parse the data used in this experiment, and by having PMPG+PCFG parse the data used in the experiments described in Bod (1999).
- The bottleneck of the sparse grammar problem prevents us from fully exploiting the disambiguating power of the pattern-matching algorithm. The GRAEL-system (GRammar Adaptation, Evolution and Learning) that is currently being developed, tries to address the problem of grammatical sparseness by using evolutionary techniques to generate, optimize and complement grammars.

10 Conclusions

Even though DOP1 exhibits outstanding parsing behavior, the efficiency of the model is rather problematic. The introduction of multiple derivations causes a considerable amount of computational overhead. Neither is it clear how the concept of multiple derivations translates to a psycholinguistic context: there is no proof that language users consider different instantiations of the same parse, when deciding on the correct analysis for a given sentence.

A pattern-matching scheme was presented that tried to disambiguate parse forests by trying to maximize the size of the substructures that can be retrieved from memory. This straightforward memory-based interpretation yields sub-standard parsing accuracy. But the combination of common-sense probabilities and enhanced context-sensitivity provides a workable parse forest disambiguator, indicating that language users might exert a complex combination of memory-based recollection techniques and stored statistical data to analyze utterances.

References

Bod, R. 1995. Enriching linguistics with statistics: Performance models of natural language. Dissertation, ILLC, Universiteit van Amsterdam.

Bod, Rens. 1999. *Beyond Grammar—An Experience-Based Theory of Language*. Cambridge, England: Cambridge University Press.

Daelemans, W., J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 1999. TiMBL: Tilburg Memory

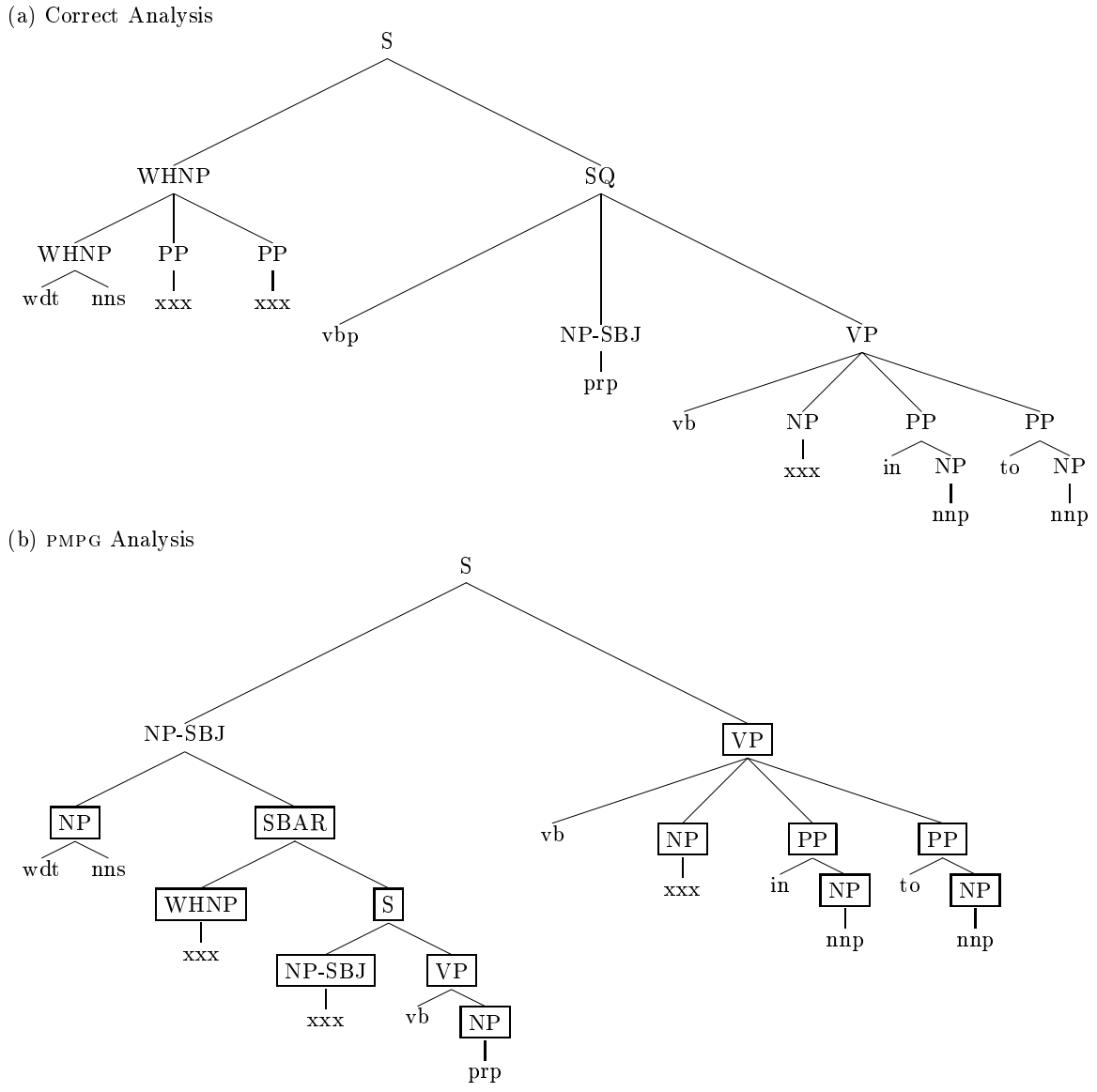


Figure 3: PMPG Error Analysis

Based Learner, version 2.0, reference manual. Technical Report ILK-9901, ILK, Tilburg University.

Daelemans, Walter. 1999. Memory-based language processing. *Journal for Experimental and Theoretical Artificial Intelligence*, 11:3:287–467.

De Pauw, Guy. 2000. *Probabilistische Parsers - Contextgevoeligheid en Pattern-Matching.* Antwerpen, Belgium: Antwerp Papers in Linguistics.

Goodman, Joshua. 1996. Efficient algorithms for parsing the dop model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* pages 143–152.

Marcus, M., B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Scha, R. 1990. Taaltheorie en taaltechnologie: competence en performance. In Q. A. M. de Kort and G. L. J. Leerdam, editors, *Computertoepassingen in de Neerlandistiek*, LVVN-jaarboek. Landelijke Vereniging van Neerlandici.

van Halteren, H., J. Zavrel, and W. Daelemans. 1998. Improving data-driven wordclass tagging by system combination. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics, Montr'eal, Quebec, Canada*, pages 491–497, Montreal, Canada, August 10-14.