Bootstrapping Lexicalized Models in Memory-Based Dependency Parsing

Abstract

Previous research has shown that a lexicalized parsing model incorporating words but no parts-of-speech can outperform a model involving parts-of-speech but no words given enough training data for supervised learning. We show that the same effect can be achieved with a bootstrapping approach, where a mixed model trained on a small treebank is used to parse a larger corpus which is used as training data for the lexicalized model. The results are obtained using a memory-based learning algorithm for deterministic dependency parsing of unrestricted Swedish text.

1 Introduction

The role of part-of-speech tagging in data-driven approaches to syntactic parsing is a moot point. In early work on treebank parsing it was more or less standard practice to have a separate tagging phase prior to parsing proper (Charniak, 1996), but with the emergence of lexicalized models it was found that better parsing accuracy could be obtained if the part-of-speech analysis was integrated in the parsing process (Charniak, 1997; Collins, 1997). More recently, it has been argued that the main reason for using parts-of-speech in data-driven parsing is that they provide a back-off model for lexical items and thereby counteract the sparse data problem (Charniak, 2000; van den Bosch and Buchholz, 2002).

In a study of memory-based shallow parsing, Van den Bosch and Buchholz (2002) showed that a model incorporating words but no parts-of-speech, while inferior with small training data sets, outperforms a model involving parts-of-speech but no words for training sets over a certain size (which in their experiments was around 50 000 sentences). In principle, these results indicate that we could eliminate one bottleneck in data-driven parsing by using pure lexical models trained on very large data sets, instead of having to rely on training data with part-of-speech tagging. In practice, however, it is difficult to cash in on this idea, at least for treebank parsing using supervised learning, because we are normally faced with an even tighter bottleneck given by the limited availability of treebank data. For example, for Swedish, the largest corpus with manually corrected part-of-speech tagging is on the order of 1 million tokens (Ejerhed and Källgren, 1997), whereas the largest treebank is only one tenth of that size (Nivre, 2002).

In this paper, we investigate whether it is possible to bootstrap the lexical models, so that we can start from a small treebank and still obtain the positive effects of having a lexical model trained on a large dataset. Like van den Bosch and Buchholz (2002), we use a memory-based approach to learning, but the task considered is deterministic dependency parsing (Yamada and Matsumoto, 2003; Nivre, 2003), rather than shallow parsing in the traditional sense. The paper is structured as follows. Section 2 presents our general approach to deterministic dependency parsing, and section 3 explains how memory-based learning can be used

to guide the parser. The experimental setup is described in section 4, the results are presented in section 5, and conclusions are stated in section 6.

Deterministic dependency parsing has been proposed as a robust and efficient method for syntactic parsing that combines properties of deep and shallow parsing (Yamada and Matsumoto, 2003; Nivre, 2003). Dependency parsing means that the goal of the parsing process is to construct a labeled dependency graph of the kind depicted in Figure 1. Deterministic parsing means that we derive a single analysis for each input string, with no redundancy or backtracking, which makes it possible to parse sentences in linear time (Nivre, 2003).

2 Deterministic Dependency Parsing

The parsing algorithm used is the one presented in Nivre (2003), which is in many ways similar to the basic shift-reduce algorithm for context-free grammars (Aho et al., 1986), although the parse actions are different given that no nonterminal symbols are used. Moreover, unlike the algorithm of Yamada and Matsumoto (2003), the algorithm considered here actually uses a blend of bottom-up and top-down processing, constructing left-dependencies bottom-up and right-dependencies top-down, in order to achieve incrementality. For a similar but nondeterministic approach to dependency parsing, see Obrebski (2003).

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 $\langle \mathbf{nil}, W, \emptyset \rangle$ and terminates when it reaches a configuration $\langle S, \mathbf{nil}, A \rangle$ (for any list S and set of arcs A). The behavior of the parser is defined by the transitions defined in Figure 2, where w_i, w_j and w_k are arbitrary word tokens, and r and r' are arbitrary dependency types (arc labels):

1. The transition **Left-Arc** (**LA**) adds an arc $w_j \xrightarrow{r} w_i$ from the next input token w_j to the token w_i on top of the stack and reduces (pops) w_i from the stack.
2. The transition **Right-Arc** (**RA**) adds an arc $w_i \xrightarrow{r} w_j$ from the token w_i on top of the stack to the next input token w_j, and shifts (pushes) w_j onto the stack.
3. The transition **Reduce** (**RE**) reduces (pops) the token w_i on top of the stack.
4. The transition **Shift** (**SH**) shifts (pushes) the next input token n onto the stack.

The transitions **Left-Arc** and **Right-Arc** are subject to conditions that ensure that each word has at most one head in the dependency graph, while **Reduce** can only be applied if the token on top of the stack already has a head. For **Shift**, the only condition is that the input list is non-empty.

As it stands, this transition system is nondeterministic, since several transitions can often be applied to the same configuration. To get a deterministic parser, we need to introduce a mechanism for resolving transition conflicts. One way of doing this is to use a treebank to train classifiers that can predict the next transition (and dependency type) given the current configuration of the parser. In the experiments reported here, we use memory-based learning to train such 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). 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 deterministic parsing includes Venstra 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).

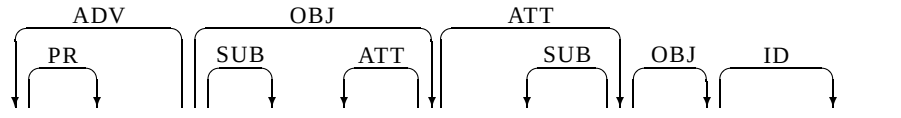

Figure 1: Dependency graph for Swedish sentence

Initialization	$\langle \mathbf{nil}, W, \emptyset \rangle$	
Termination	$\langle S, \mathbf{nil}, A \rangle$	
Left-Arc	$\langle w_i \vert S, w_j \vert I, A \rangle \rightarrow \langle S, w_j \vert I, A \cup \{(w_j, r, w_i)\} \rangle$	$\neg \exists w_k \exists r'(w_k, r', w_i) \in A$
Right-Arc	$\langle w_i \vert S, w_j \vert I, A \rangle \rightarrow \langle w_j \vert w_i \vert S, I, A \cup \{(w_i, r, w_j)\} \rangle$	$\neg \exists w_k \exists r'(w_k, r', w_j) \in A$
Reduce	$\langle w_i \vert S, I, A \rangle \rightarrow \langle S, I, A \rangle$	$\exists w_j \exists r(w_j, r, w_i) \in A$
Shift	$\langle S, w_i \vert I, A \rangle \rightarrow \langle w_i \vert S, I, A \rangle$	

Figure 2: Parser transitions

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 possible parser configurations and R is the set of dependency types. However, in order to make the problem tractable, we try to learn 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. The features used in this study are listed in Table 1.

The first five features (TOP – TOP.RIGHT) deal with properties of the token on top of the stack. In addition to the word form itself (TOP), we consider its part-of-speech, the dependency type by which it is related to its head (which may or may not be available in a given configuration depending on whether the head is to the left or to the right of the token in question), and the dependency types by which it is related to its leftmost and rightmost dependent, respectively (where the current rightmost dependent may or may not be the rightmost dependent in the complete dependency tree). The next three features (NEXT – NEXT.LEFT) refer to properties of the next input token. In this case, there are no features corresponding to TOP.DEP and TOP.RIGHT, since the relevant dependencies can never be present at decision time. The following three features (NEXT+1 – NEXT+2) are lookahead features, referring to tokens occurring immediately after NEXT in the input string, while the remaining two features (TOP−1, TOP−1) consider the token immediately below TOP on the stack.

Given the features in Table 1, we have defined three different models of parser state, one which includes word tokens but no parts-of-speech, one which includes parts-of-speech but no word tokens, and one which includes both kinds of information. All of the models in addition include dependency type information. The selection of features for each model is shown in Table 2. It should be pointed out that the **Words+PoS** model is not a simple union of the **Words** and **PoS** models, since the **Words** model includes features that are not found in any of the other models (NEXT+1, NEXT+2, TOP−1, TOP−1.DEP). The reason for this asymmetry is that preliminary experiments showed a significant improvement when these features were added to the **Words** model, whereas no improvement was found for corresponding features in the other models. Thus, the three models selected for the final experiment can be seen as the best representatives of their respective class, rather than strictly parallel counterparts.

4 Experimental Setup

Given the results of van den Bosch and Buchholz (2002), it can be expected that the **PoS** model will outperform the **Words** model when the training data set is small, as is the case for the available treebank of Swedish. However, the hypothesis that we want to test is whether the **Words** model will perform better than the **PoS** model when trained on a larger corpus, which has been parsed using a model derived from the small treebank. Therefore, the experiment involves two steps.

4.1 Step 1: Training on a Treebank

In the first step, we train each of the models on a manually annotated corpus of written Swedish, created at Lund University in the 1970's and consisting mainly of informative texts from official sources (Einarsson, 1976). Although the original annotation scheme is an eclectic combination of constituent structure, dependency structure, and topological fields (Teleman, 1974), it has proven possible to convert the annotated sentences to dependency graphs with fairly high accuracy. In the conversion process, we have reduced the original fine-grained classification of grammatical functions to a more restricted set of 16 dependency types. We refer to this training corpus as Talbanken, following Einarsson (1976), although it is also known in the literature as the Teleman corpus (Brants and Samuelsson, 1995).

Since the function we want to learn is a mapping from parser states to transitions (and dependency types), the treebank data cannot be used directly as training data. Instead, we have to simulate the parser on the treebank in order to derive, for each sentence, the transition sequence corresponding to the correct dependency graph. Given the result of

Feature	Description
TOP	The token on top of the stack
TOP.POS	The part-of-speech of TOP
TOP.DEP	The dependency type of TOP
TOP.LEFT	The dependency type of TOP's leftmost dependent
TOP.RIGHT	The dependency type of TOP's rightmost dependent
NEXT	The next input token
NEXT.POS	The part-of-speech of NEXT
NEXT.LEFT	The dependency type of NEXT's leftmost dependent
NEXT+1	The next plus one input token
NEXT+1.POS	The part-of-speech of the next plus one input token
NEXT+2	The next plus two input token
TOP−1	The token immediately below the top of the stack
TOP−1.DEP	The dependency type of the token immediately below the top of the stack

Table 1: Parser state features

Model	T	T.P	T.D	T.L	T.R	N	N.P	N.L	N+1	N+1.P	N+2	T−1	T−1.D
Words	+		+	+	+	+		+	+		+	+	+
PoS		+	+	+	+		+	+		+			
Words+PoS	+	+	+	+	+	+	+	+		+			

Table 2: Parser state models

this simulation, we can construct a data set consisting of pairs $\langle s, t \rangle$, where s is a parser state and t is the correct transition from that state (including a dependency type if applicable).

The complete converted treebank contains 6316 sentences with a mean sentence length of 15.5 words. The treebank has been divided into three non-overlapping data sets: 80% for training, 10% for validation, and 10% for final testing (random samples). The training set contains 5054 sentences, which yields a set of 371 977 training instances (transitions) for the memory-based learner.

The models have been trained using the Tilburg Memory Based Learner (TiMBL) (Daelemans et al., 2003) with the following parameter settings, which have been found to work well in earlier experiments (Nivre et al., 2004):

- The IB1 algorithm (Aha et al., 1991).
- The modified value distance metric (MVDM) (Stanfill and Waltz, 1986; Cost and Salzberg, 1993) with a frequency threshold (and the Overlap metric for lower frequencies).
- No weighting of features.
- Classification based on 5 nearest neighbors.
- Distance weighted class voting with inverse distance weighting (Dudani, 1976).

The frequency threshold for MVDM was set to 2 for the **Words** model and 3 for the **Words+PoS**. The **PoS** model turned out to be insensitive to this threshold, so the default value of 1 was used in this case. For more information about the different parameters and settings, the reader is referred to Daelemans et al. (2003).

4.2 Step 2: Training on a Parsed Corpus

In the second step, we used the three parsers derived in step 1 to parse the Stockholm-Umeå Corpus (SUC) of written Swedish (Ejerhed and Källgren, 1997), consisting of 73 199 sentences (1 152 473 tokens). This gave us three parsed corpora, which could be used to derive three sets each consisting of 2 033 792 transitions, which were added to the original training set from Talbanken

to give three new training sets, which we will call **T-Words**, **T-PoS** and **T-Words+PoS**, respectively.

Although we are primarily interested in the comparison of **Words** and **PoS**, and although retraining a model on its own output is unlikely to improve its performance, we have retrained all three models on all three new training sets, which gives a total set of nine parsers to compare with the three parsers constructed in step 1. No new parameter optimization was performed, which means that each model was retrained with the same parameter settings as in step 1. For the models **PoS** and **Words+PoS**, the gold standard part-of-speech annotation in SUC was used for features involving parts-of-speech.

4.3 Evaluation

In evaluating the performance of the different parsers constructed in the experiment, we have used the validation data set from Talbanken, whereas the final test set has not been used at all in the work reported in this paper (cf. section 4.1). The evaluation metric used is the attachment score (AS), which is a standard measure used in studies of dependency parsing (Eisner, 1996b; Collins et al., 1999). The attachment score is computed as the proportion of tokens in a sentence (excluding punctuation) that are assigned the correct head (or no head if the token is a root). The overall attachment score is then calculated as the mean attachment score over all sentences in the sample. In order to measure the accuracy for dependency types, we have also defined a labeled attachment score (LAS), where both the head and the label (dependency type) must be correct, but which is otherwise computed in the same way as the ordinary (unlabeled) attachment score. For statistical significance testing we use a one-way ANOVA for correlated samples.

5 Results

Table 3 shows the accuracy of the three parsers trained in the first step. As expected, the **PoS** model achieves a better accuracy than the **Words** model, both with respect to labeled and unlabeled attachment. However, both models are clearly outperformed by the **Words+PoS** model. This confirms that lexical information is crucial for parsing accuracy, as shown in many previous studies (Eisner, 1996b; Charniak, 1997; Collins, 1997), but also that parts-of-speech are more reliable than words alone with sparse data, as shown by van den Bosch and Buchholz (2002).

There are no comparable results available for dependency parsing of Swedish, but if we compare with results for other languages, we find that the unlabeled attachment score is slightly lower than the best published results for English, which are just above 90% (Eisner, 1996a; Collins et al., 1999), but substantially higher than the results for Czech (Collins et al., 1999).

Table 4 shows the results for the models trained on the three parsed versions of the Stockholm-Umeå Corpus. If we begin by considering the **Words** model, we see a significant improvement in accuracy, both labeled and unlabeled, when this model is trained on the output of either of the two other models. The improvement is greater for **T-Words+PoS**, which is natural given that this version of the parsed corpus should be more accurate, but the difference between **T-Words+PoS** and **T-PoS** is only significant for labeled attachment score. When the **Words** model is trained on its own output, there is a small drop in accuracy, but this is not statistically significant.

If we turn to the **PoS** model, we see a significant drop in accuracy when this model is trained on **T-Words**. For the two other conditions, there are no significant differences, which again indicates that a model based on parts-of-speech alone reaches a stable state with relatively little training data but has a limited potential for improvement.

If we compare the **Words** and **PoS** models, we note that while the **PoS** model has significantly better accuracy when trained on the small treebank, the **Words** model has equal or better accuracy in all cases after bootstrapping. (The small negative differences with respect to **T-PoS** are not statistically significant.) As already noted, the best results are obtained when training on **T-Words+PoS**, where especially the labeled attachment score for **Words** is substantially higher than for **PoS** and in fact approaches the results for **Words+PoS**. With respect to unlabeled accuracy, the difference between **Words** and **PoS** is not statistically significant.

Model	LAS	AS
Words	77.0	83.8
PoS	78.7	85.2
Words+PoS	83.9	88.0

Table 3: Parsing accuracy for models trained on Talbanken

Training set	Model	LAS	AS
T-Words	Words	76.8	83.1
	PoS	74.1	81.7
	Words+PoS	78.0	83.6
T-PoS	Words	78.5	84.9
	PoS	78.6	85.3
	Words+PoS	79.3	85.7
T-Words+PoS	Words	81.1	85.8
	PoS	79.0	85.2
	Words+PoS	83.2	87.1

Table 4: Parsing accuracy for models trained on the parsed Stockholm-Umeå Corpus

If we turn to the **Words+PoS** model, finally, we see that this model never improves with bootstrapping. When trained on its own output the drop in accuracy is not significant, but when trained on the output of the other models the accuracy drops especially with respect to labeled attachment. And the highest scores overall are the ones obtained with the **Words+PoS** model trained only on Talbanken. This may seem like a disappointing result, but we have to keep in mind that whereas the best model so far is restricted by the bottleneck of available treebank data, the bootstrapping of the pure lexical model can be performed with even larger training corpora. As far as we know, it is still an open question whether it is actually possible in this way to surpass the accuracy of the model on which the bootstrapping is based.

6 Conclusion

In this paper, we have shown that a pure lexicalized model for deterministic dependency parsing can give a higher accuracy than a model involving (only) parts-of-speech, if we use a mixed model trained on a small treebank to generate a larger training corpus for the lexicalized model. In this way, we have extended the results of van den Bosch and Buchholz (2002) by showing that, not only can a parsing model based on words alone outperform a model based on parts-of-speech alone given enough training data, but this effect can also be achieved through bootstrapping, starting from a relatively small initial training corpus. This opens up new possibilities, since the amount of training data that can be used for the lexical model is no longer bounded by the amount of annotated treebank data available, even if supervised learning is used. One of the interesting questions for further research is whether we can use this potential to surpass the best model trained on the initial treebank.

Acknowledgements

The work presented in this paper was supported by a grant from the Swedish Research Council (621-2002-4207). The memory-based classifiers used in the experiments were constructed using the Tilburg Memory-Based Learner (TiMBL) (Daelemans et al., 2003).

References

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

Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. 1986. *Compilers: Principles Techniques, and Tools*. Addison Wesley.

Thorsten Brants and Christer Samuelsson. 1995. Tagging the Teleman corpus. In Kimmo Koskenniemi, editor, *Proceedings of the 10th Nordic Conference of Computational Linguistics (NODALIDA)*, pages 7–20.

Eugene Charniak. 1996. Tree-bank grammars. In *AAAI/IAAI, Vol. 2*, pages 1031–1036.

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

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

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

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

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

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

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2003. Timbl: Tilburg memory based learner, version 5.0, reference guide. Technical Report ILK 03-10, Tilburg University, ILK.

Walter 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.

Jan Einarsson. 1976. Talbankens skriftspråkskonkordans. Lund University.

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

Jason M. Eisner. 1996b. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING-96*, Copenhagen.

Eva Ejerhed and Gunnel Källgren. 1997. Stockholm Umeå Corpus. Version 1.0. Produced by Department of Linguistics, Umeå University and Department of Linguistics, Stockholm University. ISBN 91-7191-348-3.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. Submitted to CoNLL 04.

Joakim Nivre. 2002. What kinds of trees grow in swedish soil? a comparison of four annotation standards for swedish. In Erhard Hinrichs and Kiril Simov, editors, *Proceedings of the First Workshop on Treebanks and Linguistic Theories*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In Gertjan van Noord, editor, *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160.

Tomasz Obrebski. 2003. Dependency parsing using dependency graph. In Gertjan van Noord, editor, *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 217–218.

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

Ulf Teleman. 1974. *Manual för grammatisk beskrivning av talad och skriven svenska*. Studentlitteratur.

Antal van den Bosch and Sabine Buchholz. 2002. Shallow parsing on the basis of word only: A case study. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 443–440.

Jorn Venstra and Walter Daelemans. 2000. A memory-based alternative for connectionist shift-reduce parsing. Technical Report ILK-0012, University of Tilburg.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In Gertjan van Noord, editor, *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 195–206.