

# Multiword Units in Syntactic Parsing

Joakim Nivre and Jens Nilsson

School of Mathematics and Systems Engineering  
Växjö University  
SE-35195 Växjö, Sweden  
firstname.lastname@msi.vxu.se

## Abstract

The influence of MWU recognition on parsing accuracy is investigated with respect to a deterministic dependency parser based on memory-based learning. The effect of a perfect MWU recognizer is simulated through manual annotation of MWUs in corpus data used for training and testing two different versions of the parser, one which is lexicalized and one which is not. The results show a significant improvement in parsing accuracy not only for MWUs themselves but also with respect to surrounding syntactic structures. The improvement is consistent for both versions of the parser and holds for both labeled and unlabeled accuracy. The greatest improvement corresponds to a 5% error reduction, which is substantial given the relatively low frequency of MWUs in the data. In addition to a quantitative analysis, we also present an analysis of some of the most important error types that are eliminated by the recognition of MWUs.

## 1. Introduction

The relationship between multiword units (MWUs) and syntactic parsing is a complex one involving several related but distinct questions:

1. Should MWUs be recognized as such and given a special treatment in the syntactic analysis?
2. If the answer is *yes*, is this analysis best performed by the parser itself or by a specialized module applying either before or after the parsing process?
3. If the answer is *no*, can it nevertheless be beneficial to recognize MWUs before or during parsing, even though this is not reflected in the final analysis?

The answer to the first of these questions is largely independent of the parsing problem itself and rather depends on the larger application in which the parsing system is embedded. Moreover, the answer may be different for different classes of MWUs. In information extraction, for example, multiword names (named entities) are highly relevant, whereas compound prepositions are generally not. Questions of this kind are outside the scope of this paper, though. Instead, we will focus on the treatment of MWUs *within* syntactic parsing.

If the first question is dependent on the larger application but largely independent of the specific parsing method used, the inverse probably holds for the second and third question. That is, whether specialized techniques for the analysis of MWUs improves parsing — either of MWUs themselves or of surrounding structures — could be highly dependent on the parsing methodology used. Many parsing frameworks do include a special treatment of multiword units, presumably on the assumption that it improves accuracy (Karlsson et al., 1995; Tapanainen, 1999). However, we have not been able to find any published study where this assumption has been tested in controlled experiments.

Although a conclusive answer to these questions would require us to study a whole range of different parsing techniques, we hope to be able to shed some light on the issues

involved by studying two different versions of a memory-based dependency parser, one which is lexicalized and one which is not. More precisely, we have conducted experiments where we simulate a perfect MWU recognizer through manual annotation and study the effect that such a component would have on parsing accuracy, both with respect to the MWUs themselves and with respect to surrounding syntactic structures. It is an open question to what extent the results can be generalized to other parsing methods, but the present study can at least be seen as a first step towards answering the more general questions.

The rest of the paper is structured as follows. First, we describe our general parsing method, which can be characterized as deterministic dependency parsing (section 2). Secondly, we explain how memory-based learning can be used to guide the deterministic parser and we describe the two different versions of the parser used in the experiments (section 3). Next, we report a series of experiments investigating the influence of MWU recognition on parsing accuracy (section 4). Finally, we state our conclusions and give directions for future research (section 5).

## 2. Deterministic Dependency Parsing

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).

In formal terms, we define dependency graphs in the following way:

1. Let  $R = \{r_1, \dots, r_m\}$  be the set of dependency types (arc labels).
2. A dependency graph for a word string  $W = w_1 \dots w_n$  is a labeled directed graph  $D = (W, A)$ , where

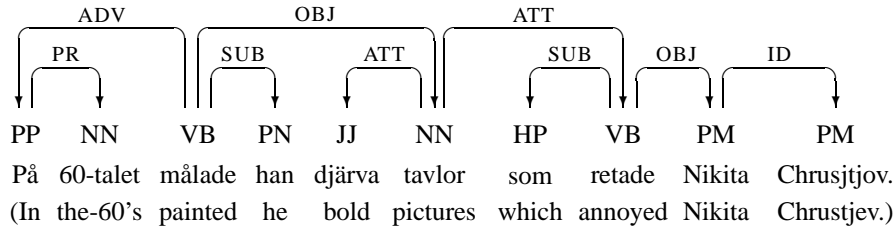


Figure 1: Dependency graph for Swedish sentence

- (a)  $W$  is the set of nodes, i.e. word tokens in the input string,
- (b)  $A$  is the arc relation, i.e. a set of labeled directed arcs  $(w_i, r, w_j)$  ( $w_i, w_j \in W, r \in R$ ),
- (c) for every node  $w_j \in W$ , 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.

The parsing algorithm used here was first defined for unlabeled dependency parsing (Nivre, 2003) and subsequently extended to labeled graphs (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  $\langle \text{nil}, W, \emptyset \rangle$  and terminates when it reaches a configuration  $\langle S, \text{nil}, A \rangle$  (for any list  $S$  and set of arcs  $A$ ). Given an arbitrary configuration of the parser, there are four possible transitions to the next configuration:

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 the token  $n$  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  $\langle n|S,I,A \rangle$ .

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).

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 the experiments reported here, we use memory-based learning to train these 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). It has also been used to guide deterministic parsers (Veenstra and Daelemans, 2000; 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:

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

Here  $\text{Config}$  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

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 (if any)
TOP.LEFT	The dependency type of TOP’s leftmost dependent (if any)
TOP.RIGHT	The dependency type of TOP’s rightmost dependent (if any)
NEXT	The next input token
NEXT.POS	The part-of-speech of NEXT
NEXT.LEFT	The dependency type of NEXT’s leftmost dependent (if any)
LOOK.POS	The part-of-speech of the next plus one input token

Table 1: Parser state features

word form itself (TOP), we consider its part-of-speech (as assigned by an automatic part-of-speech tagger in a pre-processing phase), 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 following 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 final feature (LOOK) is a simple lookahead, using the part-of-speech of the next plus one input token.

In the experiments reported below, we have used two different parser state models, one called the *lexical* model, which includes all nine features, and one called the *non-lexical* model, where the two lexical features TOP and NEXT are omitted. The learning algorithm used is the IB1 algorithm (Aha et al., 1991) with  $k = 5$ , i.e. classification based on 5 nearest neighbors.<sup>1</sup> Distances are measured using the modified value difference metric (MVDM) (Stanfill and Waltz, 1986; Cost and Salzberg, 1993), and classification is based on distance weighted class voting with inverse distance weighting (Dudani, 1976). For more information about the different parameters and settings, see Daelemans et al. (2003).

## 4. Experiments

The data used for the experiments come from 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 trees with fairly high accuracy. What makes this corpus especially suitable

<sup>1</sup>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 (Aha et al., 1991).

for this study, is the fact that several classes of MWUs have been annotated in such a way that they can be identified automatically. These expressions can be divided into three broad classes, exemplified below:

- Multiword names (MN):  
*Torsten Nilsson* (person)  
*Västra Frölunda* (place)  
*Södra Kyrkogatan* (street)
- Numerical expressions (NE):  
*3 - 4*  
*6 X 6*  
*1967 / 68*
- Compound function words, which can be subdivided into five categories:
  1. Adverbs (AB):  
*så småningom* (eventually)  
*i allmänhet* (in general)  
*var som helst* (anywhere)
  2. Prepositions (PP):  
*på grund av* (because of)  
*med hänsyn till* (with respect to)  
*i samband med* (in connection with)
  3. Subordinating conjunctions (SN):  
*efter det att* (after)  
*även om* (even if)  
*trots att* (despite the fact that)
  4. Determiners (DT):  
*den här* (this)  
*en och samma* (one and the same)  
*en del* (some)
  5. Pronouns (PN):  
*den här* (this)  
*var och en* (everyone)  
*en del* (some)

Table 2 gives frequency counts for different categories of MWUs in the corpus and compares them to the overall statistics for tokens and sentences. We see that compound function words, taken together, is the largest class, followed by multiword names, whereas numerical expressions are rare in the corpus. The overall frequency of the annotated MWUs can be appreciated by noting that, on average, there

Category	Example	Frequency
MN	Torsten Nilsson	427
NE	3 - 4	10
AB	så småningom (eventually)	826
PP	på grund av (because of)	256
SN	efter det att (after)	167
DT	den här (this)	94
PN	den här (this)	30
MWUs		1810
Sentences		6316
Tokens		97623

Table 2: Statistics of the data set

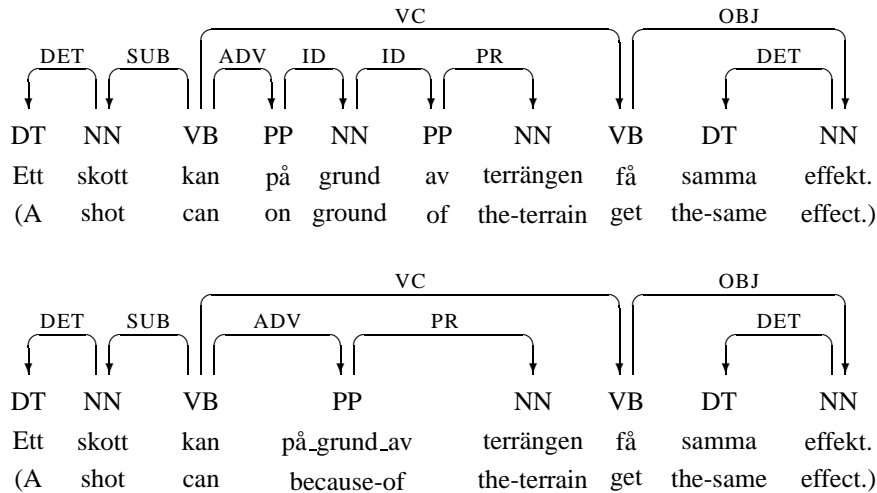


Figure 2: Treebank representations: Baseline (top) and MWU (bottom)

are 3 MWUs in 10 sentences and 2 MWUs in 100 words. It should be emphasized, however, that this only takes into account the MWUs that could be identified through the original corpus annotation. However, the overall frequency figures accord well with results from automatic acquisition of MWUs with reported extraction rates of 1–3 MWUs per 100 words (Dias et al., 1999).

For the experiments we have created two different versions of the treebank. In the baseline version, each MWU is represented as a sequence of tokens, each with its own part-of-speech, combined into a left-headed dependency chain with arcs labeled with the special symbol ID. In the MWU version, each MWU is represented by a single token, which is formed by joining the constituent tokens with underscores (e.g. *på grund av* → *på\_grund\_av*) and assigned a part-of-speech appropriate for the entire MWU.

By way of illustration, Figure 2 shows one and the same sentence in both representations. The experiments consist in training and testing the memory-based dependency parser on the two different versions of the treebank, measuring the parsing accuracy both for MWUs themselves (under the baseline condition) and for all other syntactic structures (both conditions). All experiments have been carried out with both the lexical and the non-lexical versions of the parser.

For other experiments, the treebank has been divided into three non-overlapping data sets: 80% for training 10% for development/validation, and 10% for final testing (random samples). The results presented in this study are all from the validation set, i.e. all parsers have been trained on the training set and evaluated on the validation set. (The final test set has not been used at all in the experiments reported in this paper.)

Parsing accuracy is measured in the experiments by three different metrics:

- **Attachment score (AS):** The proportion of tokens (excluding punctuation) that are assigned the correct head (or no head if the token is a root) (Eisner, 1996; Collins et al., 1999).
- **Labeled attachment score (LAS):** The proportion of tokens (excluding punctuation) that are assigned both the correct head and the correct dependency type (or no head if the token is a root).
- **MWU accuracy:** Labeled precision, recall and  $F_{\beta=1}$  score for the special ID relation (only applicable under the baseline condition).

Table 3 shows the accuracy with which the parsers can learn to recognize MWUs by combining their component

Model	Prec	Rec	$F_{\beta=1}$
Non-lexical	55.6	27.4	36.7
Lexical	75.7	67.1	71.1

Table 3: Parsing accuracy for MWUs only

word tokens into left-headed dependency chains labeled with the dedicated ID relation. We see that none of the parsers trained on the baseline data achieve a very high accuracy. Not surprisingly, the lexicalized parser performs best, with 75.7% precision and 67.1% recall for the ID relation, while the non-lexicalized parser only reaches 55.6% precision and 27.4% recall. Although these results are perhaps not very interesting in themselves, we can at least take the performance of the lexicalized parser, corresponding to an  $F_{\beta=1}$  score of 71.1%, as a baseline for further experiments with fully automatic MWU recognition. If the overall accuracy is going to improve, a dedicated MWU recognizer will at least have to do better than the parser on its own.

Table 4 compares the accuracy of the parsers when trained and tested on the two versions of the data set. We see that both parsers perform consistently better with the MWU representation than with the baseline representation, even if errors in the MWUs themselves are not counted. With respect to the basic attachment score, there is an improvement from 83.0% to 83.5% for the non-lexical parser, and from 84.7% to 85.3% for the lexical one. With respect to labeled attachment, the improvement is even greater. The non-lexical parser goes from 76.1% to 77.4%, and the lexical parser from 80.7% to 81.6%. All differences are significant beyond the 0.0001 level (McNemar’s test).

The experimental results show unequivocally that the recognition of MWUs improves parsing accuracy not only with respect to the MWUs themselves, but also with respect to surrounding syntactic structures. The differences are quantitatively fairly small, and the largest gain only corresponds to a 5% error reduction, but we have to keep in mind that MWUs are relatively rare themselves. We noted earlier that there are about 2 MWUs per 100 words. So, if parsing accuracy improves by 1% then, roughly speaking, one MWU in two makes a difference. It is therefore not surprising that the differences are highly statistically significant, even though they are numerically small compared to the total number of parsing errors.

On the other hand, we must also remember that the MWU representation used in the experiments is a simulation of 100% accurate MWU recognition, which is of course impossible to achieve in practice. It remains to be seen how much of the theoretically possible improvement can be realized when using automatic methods for MWU recognition.

If we look in detail at the kind of parsing errors that are eliminated under the MWU condition, they can be divided into two broad classes:

- Errors that occur in the vicinity of a MWU and where the baseline parsers are misled either by an incorrect analysis of the MWU or by the fact that the MWU is not recognized as a lexical unit. We will call these MIC errors (for “MWU in context”).
- Errors that do not occur in the vicinity of a MWU but which are due to “noise” in the training data resulting from the failure to recognize MWUs as lexical units. We will call these MOC errors (for “MWU out of context”).

MIC errors can be further subdivided into several types, and although an exhaustive analysis of these errors is beyond the scope of this paper, we will discuss two of the most important types here. The first type occurs when the internal structure of the MWU is anomalous given its syntactic function as a unit. A good example is found in Figure 3, involving the MWU *vad beträffar* (as regards), where the second word token is a finite verb. Thus, the lexical baseline parser analyzes this as a unit with clause structure, treating the *wh*-pronoun *vad* (what) as the subject of the verb *beträffar* (regards), and the following conjoined noun phrase *jordbruk och industri* (agriculture and industry) as the object of this verb. By contrast, the lexical MWU parser recognizes *vad beträffar* as a compound preposition, which yields a correct analysis of the entire structure.

The second type of error occurs when the internal structure of the MWU is compatible with its syntactic function, but where the fact that it is not recognized as a lexical unit results in an incorrect analysis of the surrounding structure, typically in the form of an attachment error. Figure 4 gives an example involving the compound adverb *i regel* (as a rule), which is analyzed as an ordinary prepositional phrase by the lexical baseline parser. Now, this does not lead to an incorrect analysis of its syntactic function, since the adverbial function is perfectly compatible with a prepositional phrase analysis. However, it leads to an incorrect attachment decision in that the following agent phrase *av kommuner* (by communes) is attached as an attribute to the noun *regel* (rule), instead of being treated as an adverbial to the passive verb *drivs* (is run). Again, the MWU parser avoids this problem, because the unit *i regel* is not only assigned an adverbial function but is also treated as a compound adverb.

In the foregoing example, the error consists in making an incorrect attachment *to* the MWU, but it is also common for the baseline parsers to make incorrect attachments *of* MWUs, where the corresponding MWU parsers make correct attachments. A rather amusing example is the following:

Condition	Non-lexical		Lexical	
	AS	LAS	AS	LAS
Baseline	83.0	76.1	84.7	80.7
MWU	83.5	77.4	85.3	81.6

Table 4: Parsing accuracy excluding MWUs

Vi skulle ha stoppat dem i kassetter i stället för att slicka igen kuvert.

**Baseline:** We should have put them into cassettes in the rack in order to seal envelopes.

**MWU:** We should have put them into cassettes instead of sealing envelopes.

While the lexical MWU parser correctly analyzes *i stället för* (instead of) as a compound preposition introducing an adverbial modifier, the baseline parser treats *i stället* (in the place) as a prepositional phrase, attaching it as a post-modifier to the noun *kassetter* (cassettes). However, since the word form *stället* is ambiguous and can also mean “the rack”, and since *för att* can be analyzed as a compound subordinating conjunction (in order to), it is actually possible to derive a coherent (but contextually incorrect) analysis of the sentence where *i stället* is indeed a prepositional phrase, possibly acting as a nominal post-modifier.

MOC errors are harder to categorize than MIC errors since they are more indirectly related to the analysis of MWUs as such. But we can illustrate the phenomenon with a case that is fairly frequent in the Swedish data, namely the two-word sequence *så att*. This can either be a multi-word subordinating conjunction with a consecutive meaning (corresponding to *so that* in English), or the adverb *så* (so, like that) followed by either the subordinating conjunction *att* (that) or the infinitive marker *att* (to). Now, when trained on a corpus where these two cases are only distinguished through their syntactic analysis and not as one versus two lexical units, the parser will have to learn to disambiguate them syntactically. This will typically result in errors going in both directions, i.e. not only MWUs being analyzed as syntactic combinations but also syntactic combinations being analyzed as MWUs. And whereas the former type of error can be eliminated by recognizing MWUs prior to parsing new input, the latter type will remain unless the same MWU recognition is also applied to the training data for the parser.

Finally, it may be worth noting that the recognition of MWUs also improves the robustness of the parsers. Thus, while the lexical baseline parser fails to produce a well-formed dependency graph (i.e. a complete projective tree) for 13.3% of the sentences in the test data set, the corresponding figure for the lexical MWU parser is only 10.8%. (The corresponding figures for the non-lexical parser are 8.8% and 7.8%.) An inspection of the problematic sentences reveals that accuracy and robustness go hand in hand in this case. We have seen examples above where the baseline parsers make the wrong attachment because they fail to recognize MWUs properly. In the case of incomplete sentences, the consequence is instead that the parsers make

*no* attachment. But the cause of the problem is the same in both cases.

To sum up, we have seen that the recognition of MWUs can improve the quality of data-driven parsing in several different ways. First of all, it may improve the consistency of the training data, by eliminating word sequences that can be interpreted both as MWUs and as syntactic combinations, which leads to an improvement in overall parsing accuracy. Secondly, it can eliminate parsing errors both within MWUs and in their relation to surrounding syntactic structures. Finally, the gain in accuracy may also lead to an improvement in robustness.

## 5. Conclusion

In this paper, we have investigated the influence of MWU recognition on parsing accuracy for a deterministic dependency parser based on memory-based learning. We have simulated the effect of a perfect MWU recognizer through manual annotation of MWUs, in particular multi-word names and compound function words, in corpus data used for training and testing two different versions of the parser, one lexicalized and one non-lexicalized.

The results show a significant improvement in parsing accuracy not only for MWUs themselves, but also with respect to surrounding syntactic structures. The improvement holds for both labeled and unlabeled attachment accuracy and is consistent for both versions of the parser. The greatest improvement (labeled attachment for the non-lexical parser) corresponds to a 5% error reduction, which is substantial given the relatively low frequency of MWUs in the data. In addition to the quantitative analysis, we have also presented an analysis of some of the most important error types that are eliminated by the recognition of MWUs, making a distinction between errors that occur in the context of MWUs (MIC errors) and errors that occur in other contexts but are due to the presence of unanalyzed MWUs in the training data (MOC errors).

The results presented in this study can be taken as an indication of the maximum gain in parsing accuracy that can be achieved by using MWU recognition prior to training and parsing, at least when restricted to the MWU categories represented in our data. An important topic for future research is to see how much of this potential can be realized in practice, when relying on automatic recognition of MWUs rather than manually annotated corpus data. However, it should also be remembered that since the original corpus annotation in our case was done for a different purpose than facilitating syntactic parsing, we do not know to what extent the recognized categories of MWUs are the most suitable to improve parsing performance. This may mean that the potential for improvement is actually greater than indicated by our experimental results.

Finally, it should be pointed out again that we have really only considered a special case of the general questions formulated in the introduction. What we have shown is that for a particular kind of parser, a deterministic dependency parser guided by memory-based learning, there is a small but significant gain in accuracy that results from recognizing MWUs prior to training and parsing. It is an open question how far these results can be generalized to other parsing methods, in particular methods that are grammar-based rather than data-driven.

## 6. 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).

## 7. References

- Aha, D. (ed.), 1997. *Lazy Learning*. Kluwer.
- Aha, D. W., D. Kibler, and M. Albert, 1991. Instance-based learning algorithms. *Machine Learning*, 6:37–66.
- Collins, Michael, Jan Hajič, Eric Brill, Lance Ramshaw, and Christoph Tillmann, 1999. A Statistical Parser of Czech. In *Proceedings of 37th ACL Conference*. University of Maryland, College Park, USA.
- Cost, S. and S. Salzberg, 1993. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.
- Daelemans, Walter, 1999. Memory-based language processing. introduction to the special issue. *Journal of Experimental and Theoretical Artificial Intelligence*, 11:287–292.
- Daelemans, Walter, Antal van den Bosch, and Jakub Zavrel, 2002. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34:11–43.
- Daelemans, Walter, 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.
- Dias, Gaël, Sylvie Guilloré, and Gabriel Lopes, 1999. Language independent automatic acquisition of rigid multiword units from unrestricted text corpora. In *Actes Traitement Automatique des Langues Naturelles*.
- Dudani, S. A., 1976. The distance-weighted  $k$ -nearest neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6:325–327.
- Einarsson, Jan, 1976. Talbankens skriftspråkskonkordans. Lund University.
- Eisner, Jason M., 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING-96*. Copenhagen.
- Fix, E. and J. Hodges, 1952. Discriminatory analysis: Nonparametric discrimination: Consistency properties. Technical Report 11, USAF School of Aviation Medicine, Randolph Field, Texas.
- Karlsson, Fred, Atro Voutilainen, Juha Heikkilä, and Arto Anttila (eds.), 1995. *Constraint Grammar: A language-independent system for parsing unrestricted text*. Number Volume 4 in Natural Language Processing. Mouton de Gruyter.
- Nivre, Joakim, 2003. An efficient algorithm for projective dependency parsing. In Gertjan van Noord (ed.), *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*.
- Nivre, Joakim, Johan Hall, and Jens Nilsson, 2004. Memory-based dependency parsing. In *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL)*.
- Skousen, Royal, 1989. *Analogical Modeling of Language*. Kluwer.
- Skousen, Royal, 1992. *Analogy and Structure*. Kluwer.
- Stanfill, C. and D. Waltz, 1986. Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1228.
- Tapanainen, Pasi, 1999. *Parsing in Two Frameworks: Finite-State and Functional Dependency Grammar*. Ph.D. thesis, University of Helsinki.
- Teleman, Ulf, 1974. *Manual för grammatisk beskrivning av talad och skriven svenska*. Studentlitteratur.
- Veenstra, Jorn and Walter Daelemans, 2000. A memory-based alternative for connectionist shift-reduce parsing. Technical Report ILK-0012, University of Tilburg.
- Yamada, Hiroyasu and Yuji Matsumoto, 2003. Statistical dependency analysis with support vector machines. In Gertjan van Noord (ed.), *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*.

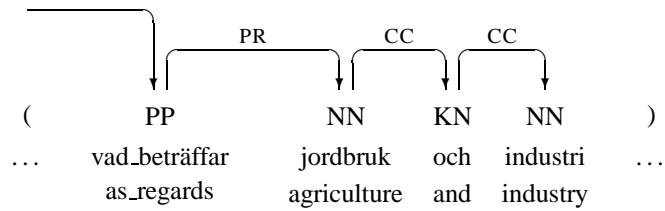
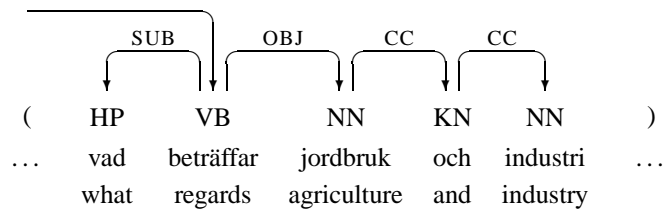


Figure 3: Syntactically anomalous MWU: Baseline (top) and MWU (bottom)

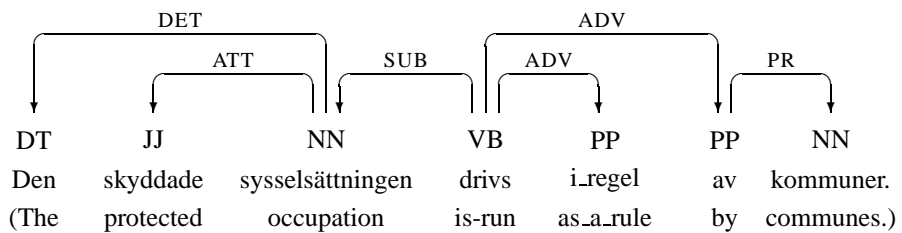
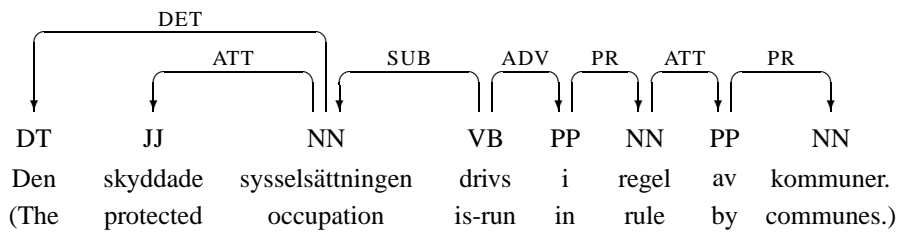


Figure 4: Attachment error: Baseline (top) and MWU (bottom)