5

Treebank Parsing

The problem of parsing unrestricted natural language text has been defined in this study as the problem of assigning to each sentence in a text its correct syntactic analysis. Conceived in this fashion, text parsing is essentially an empirical problem and the accuracy of a text parsing system can only be evaluated by comparing the analysis produced by the system to some kind of gold standard. The standard method for carrying out this kind of evaluation is to apply the system to a sample of text taken from a treebank, i.e., from a corpus where each sentence is annotated with its correct analysis. In the data-driven approach to text parsing, treebank data may also be used in the training corpus, i.e., in the sample of text on which we base our inductive inference. Using treebank data for training and evaluation is what we normally understand by the term *treebank parsing*.

This chapter presents an experimental evaluation of inductive dependency parsing based on treebank parsing. We use treebank data to train parser guides, as described in the preceding chapter, and we use treebank data to evaluate the quality of the resulting parsers, with respect to accuracy as well as efficiency. Before we turn to the evaluation, we briefly discuss treebanks and their use in research on syntactic parsing more generally, touching on some of the methodological problems that arise in using treebank data for parser evaluation. We then describe our experimental methodology, including the data sets used, the models and algorithms evaluated, and the evaluation metrics used to assess performance. The main part of the chapter is devoted to the presentation and discussion of experimental results, focusing on the influence of different kinds of parameters related to the feature model and to the learning algorithm. We conclude the chapter with a final evaluation of the best models and a comparison with related results in the literature.

5.1 Treebanks and Parsing

A treebank can be defined as a linguistically annotated corpus that includes some kind of syntactic analysis over and above part-of-speech tagging. The term *treebank* appears to have been coined by Geoffrey Leech (Sampson, 2003) and obviously alludes to the fact that the most common way of representing the syntactic analysis is by means of a tree structure. However, in current usage, the term is in no way restricted to corpora annotated with tree-shaped representations, but applies to all kinds of syntactically analyzed corpora (Abeillé, 2003a; Nivre, forthcoming).

Treebanks have been around in some shape or form at least since the 1970s. One of the earliest efforts to produce a syntactically annotated corpus was made by Ulf Teleman and colleagues at Lund University, resulting in more than 300,000 words of both written and spoken Swedish, annotated manually with grammatical functions and a limited form of phrase structure, an impressive achievement at the time but unfortunately documented only in Swedish (Teleman, 1974; Einarsson, 1976a,b; Nivre, 2002). This treebank will be reused in the experiments below.

However, it is only in the last ten to fifteen years that treebanks have been produced on a large scale for a wide range of languages, usually by combining automatic processing with manual annotation or post-editing. A fairly representative overview of available treebanks for a number of languages can be found in Abeillé (2003b), together with a discussion of certain methodological issues. This volume is well complemented by the proceedings of the annual workshops on Treebanks and Linguistic Theories (TLT) (Hinrichs and Simov, 2002; Nivre and Hinrichs, 2003; Kübler et al., 2004; Civit et al., 2005).

While corpus linguistics provided most of the early motivation for developing treebanks and continues to be one of the most important usage areas, the use of treebanks in natural language parsing has increased dramatically in recent years and has probably become the primary driving force behind the development of new treebanks. Broadly speaking, we can distinguish two main uses of treebanks in this area. The first is the use of treebank data in the evaluation of syntactic parsers, which will be discussed in section 5.1.1. The second is the application of inductive machine learning to treebank data, exemplified by the majority of data-driven approaches to text parsing. These two uses are in principle independent of each other, and the use of treebank data in evaluation is not limited to data-driven parsing systems. However, the data-driven method for research and development normally involves an iterative training-evaluation cycle, which makes not only inductive inference but also empirical evaluation an integral part of the methodology. This gives rise to certain methodological problems, which will be treated in section 5.1.2. Finally, in section 5.1.3 we will address the specific requirements on treebank data for dependency parsing.

5.1.1 Treebank Evaluation

Empirical evaluation of systems and components for natural language processing is currently a very active field. With respect to syntactic parsing there are essentially two types of data that are used for evaluation. On the one hand, we have so-called *test suites*, i.e., collections of sentences that are compiled in order to cover a particular range of syntactic phenomena without consideration of their frequency of occurrence (Lehmann et al., 1996). On the other hand, we have treebank samples, which are extracted to be representative with respect to the frequency of different phenomena. Both types of data are clearly relevant for the evaluation of syntactic parsers, but it is also clear that the resulting evaluation will focus on different properties. Test suite evaluation typically measures the coverage of a syntactic parser in terms of the number of constructions that it can handle, without considering the relative frequency of these constructions, although it can also give diagnostic information on issues such as overgeneration and overacceptance (cf. Oepen and Flickinger, 1998). Treebank evaluation, on the other hand, measures the average performance that we can expect from the parser when applied to naturally distributed data from the same source as the evaluation corpus. Given the view of text parsing adopted in this study, it is clear that treebank evaluation is the most relevant form of empirical evaluation.

Parser evaluation may focus on several different dimensions. For instance, robustness (or coverage) can be evaluated by calculating how large a proportion of the input sentences receive an analysis, and disambiguation (or leakage) can be evaluated by computing the average number of analyses assigned to a sentence, normalized with respect to sentence length (Black et al., 1993). For this kind of evaluation it is not even necessary to have annotated treebank data. However, as noted by Carroll et al. (1998), these measures are very weak in themselves, unless they are complemented by some kind of qualitative evaluation of the analyses assigned to a given sentence. For the investigations in this book, they are even less interesting, since our parsing methods guarantee exactly one analysis per sentence for any input text.

Another dimension that can in principle be evaluated without annotated treebank data is efficiency. Measuring time or memory consumption during parsing and relating it to the size of the input only requires a sample of text. Again, however, it is clear that this is a very weak form of evaluation, unless it is combined with an assessment of analysis quality. In the case of dependency parsing, it is trivial to construct an optimally efficient parser that simply analyzes each word as a dependent of the preceding word.

This brings us to the evaluation of accuracy, which is clearly the most important aspect of treebank evaluation. First, as we have just seen, it is often a necessary condition for the interpretation of other forms of evaluation. In most cases, it is simply not meaningful to compare two systems with respect to robustness, disambiguation or efficiency unless we have some way of comparing their respective accuracy. More importantly, however, the notion of empirical

accuracy is at the very heart of the notion of text parsing, as defined in this study. Whereas grammar parsing can be evaluated in terms of formal notions such as consistency and completeness, there is simply no alternative to an empirical evaluation of accuracy for text parsing. And the standard methodology for this kind of evaluation is to use a sample of treebank data as an empirical gold standard.

The basic idea is straightforward. If the treebank sample is representative of the text language that we want to analyze, then applying the parser to the text and comparing the output of the parser to the original annotation will allow us to estimate the average accuracy of the parser when applied to an arbitrary text taken from the same population. In the same fashion, comparing the output of two different parsers applied to the same sample should allow us to test the hypothesis that their average accuracy is different. On the face of it, this is a standard application of statistical inference to experimental data. In reality, there are a number of problems that arise in connection with this evaluation method, problems related to data selection, to treebank annotation and to evaluation metrics.

Starting with the problems of data selection, it is worth remembering that any application of statistical inference is based on the assumption that we have a random sample of the variable under consideration, or at least a set of independent and identically distributed (i.i.d.) variables (Lindgren, 1993). To what extent a treebank sample satisfies these conditions depends on a number of factors, some of which are not under the control of the researcher wishing to perform evaluation, such as the sampling procedure used when collecting the data for the treebank in the first place. Even if data for the treebank has been collected by means of a sampling procedure, this sampling is usually performed on the level of text blocks, such as documents or paragraphs, which means that the sampling conditions are not satisfied on the level of individual sentences. This problem becomes even more serious if we measure accuracy on the word or phrase level, as is the case for many accuracy metrics, since it is quite obvious that the individual words or phrases of a single sentence are not i.i.d. variables. By taking these factors into account when selecting data for evaluation, we can mitigate the effects of statistical dependence between measurements and avoid the overestimation of statistical significance, even if we can never in practice attain the ideal situation of having a strict random sample.

Besides problems having to do with the sampling procedure itself, we must also ask ourselves which population we are sampling from. Throughout this study we have referred to the problem of parsing unrestricted natural language text, but it is highly questionable whether we can ever sample this population by selecting data from existing treebanks. Many treebanks are limited to a single genre of text, usually newspaper text which is easily accessible, with the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993) being the most well-known example in parser evaluation. And even so-called balanced corpora usually draw their data from a limited set of text types,

often including newspaper text, literary works, and various forms of technical and scientific writing, as in the influential Brown Corpus (Kucera and Francis, 1967). Although this is not literally *unrestricted* natural language text, it is still the best we can get if we want to perform a statistical evaluation of accuracy in text parsing. But we have to keep in mind that our results will only be valid for the population from which our data have been sampled, and that the delimitation of this population is often far from clear-cut.

Another way in which treebank data depart from the ideal of unrestricted natural language text is that it has already been tokenized and segmented into sentences. Although our definition of text parsing presupposes that a text consists of a sequence of sentences, and that a sentence consists of a sequence of tokens, it is a non-trivial problem to segment naturally occurring text into sentences and tokens (Palmer, 2000), which means that we are bound to overestimate parsing accuracy when using test samples that are tokenized and segmented into sentences. Still, unless we are specifically interested in the influence of tokenization and sentence segmentation on parsing accuracy, this is a reasonable idealization in practice, which also has the advantage that it makes it easier to compare different parsers on exactly the same set of sentences and tokens.

A more serious problem is the role of the gold standard annotation in the evaluation process. The basic assumption in treebank evaluation of accuracy is that the gold standard provides the correct analysis for each sentence. In practice, this assumption is problematic for several reasons. First of all, any annotated corpus is bound to contain plain errors in the annotation, which means that in some cases the gold standard will provide an incorrect analysis of the sentence in question. If errors are rare and randomly distributed, this can be regarded as a minor problem.

Secondly, we may question the assumption that every sentence in a text has a single correct analysis, even relative to a fixed model of syntactic representation. Besides sentences that are genuinely ambiguous even in context, there is the problem of syntactic indeterminacy (Matthews, 1981), which means that more than one syntactic analysis may be compatible with an unambiguous semantic interpretation of a sentence and that one of these analyses therefore has to be chosen more or less arbitrarily. This may in turn lead to inconsistent annotation of the same syntactic construction, a problem that is supposed to be eliminated by detailed annotation guidelines, but which nevertheless exists in practice, as shown by Dickinson and Meurers (2003).

Finally, when using treebank data for evaluation it is often necessary to convert the annotation from one type of representation to another in order to fit the output of the parsers to be evaluated. Thus, many treebank annotation schemes include empty categories, which are normally removed when evaluating parsers that do not include such elements. Another relevant example is the conversion of constituency-based representations to dependency-based representations, which is necessary in the experiments reported below. This kind of conversion can seldom be performed with perfect accuracy, which means

that the converted annotation will contain a larger proportion of questionable analyses than the original one.

Despite all these problems, however, an annotated treebank can in most cases be regarded as a reasonable approximation to a gold standard, or at least as a sufficiently objective standard for the evaluation of accuracy in text parsing. The final methodological issue to be discussed concerning treebank evaluation is how to measure the correspondence between the output of a parser and the gold standard annotation, i.e., the choice of *evaluation metric*. Given a test sample $T_e = (x_1, \ldots, x_n)$, with the corresponding gold standard annotation $A_g = (y_1^g, \ldots, y_n^g)$ and the output $A_p = (y_1^p, \ldots, y_n^p)$ of some parser p, an obvious metric to use is the proportion of sentences where the parser output completely matches the gold standard annotation, usually referred to as the *exact match* (EM) criterion (where δ is the so-called Kronecker's δ that has value 1 if the two arguments are identical and 0 otherwise):

$$\text{EM} = \frac{1}{n} \sum_{i=1}^{n} \delta(y_i^g, y_i^p) \tag{5.1}$$

The EM metric has the advantage that the variables observed are sentences, rather than words or phrases, which makes statistical independence assumptions somewhat less problematic. At the same time, it is a rather crude metric, since an error in the analysis of a single word or constituent has exactly the same impact on the result as the failure to produce any analysis whatsoever.

Consequently, the most widely used evaluation metrics today are based on various kinds of partial correspondence between the parser output and the gold standard parse. The most well-known of these evaluation metrics are the PARSEVAL measures (Black et al., 1991; Grishman et al., 1992), which consider the number of matching constituents between the parser output and the gold standard, and which have been widely used in parser evaluation, in particular using data from the Penn Treebank. By comparing the number m of matching constituents to the number p of constituents produced by the parser and the number c of constituents in the gold standard analysis, we can measure the *bracketed precision* ($\frac{m}{p}$) and the *bracketed recall* ($\frac{m}{c}$). Only considering the bracketing has the advantage that it enables comparisons between parsers that use different sets of categories to label constituents. However, if constituent labels are also taken into account, we get *labeled precision* and *labeled recall* instead. Finally, it is common to include statistics on the mean number of *crossing brackets* per sentence (or the proportion of sentences that have zero crossing brackets), where a crossing bracket occurs if a parser constituent overlaps a gold standard constituent without one being properly contained in the other.

Although the PARSEVAL measures make very few assumptions about the form of syntactic representations, they do presuppose that representations are constituency-based. For dependency-based representations, the closest correspondent to these metrics is the *attachment score* (AS) (Eisner, 1996a,b;

Collins et al., 1999), which measures the proportion of words in a sentence that are attached to the correct head according to the gold standard. If we let h_g denote the gold standard assignment of dependents to heads for the sentence $x = (w_1, \ldots, w_k)$ and let h_p denote the assignment produced by the parser p, then we can define the *unlabeled* attachment score (AS_U) of p with respect to x as follows:

$$\mathrm{AS}_U(x) = \frac{1}{k} \sum_{i=1}^{k} \delta(h_g(i), h_p(i)) \tag{5.2}$$

If we also take dependency labels into account, as proposed by Lin (1995), we get a *labeled* version of the attachment score (AS_L), which is applicable to parsers that produce labeled dependency graphs (using d_g and d_p for the assignment of dependency labels to words by analogy with h_g and h_p):

$$\mathrm{AS}_L(x) = \frac{1}{k} \sum_{i=1}^{k} \delta(h_g(i), h_p(i)) \cdot \delta(d_g(i), d_p(i)) \tag{5.3}$$

When calculating the attachment score for the entire test sample, we may either calculate the mean per sentence (sometimes called the *macro-average*) or the mean per word (the *micro-average*). Although it can be argued that the former is more natural, the latter is more common in the literature. One good reason for this is that the sentence score for very short sentences can only assume a discrete set of values, which may distort the overall scores.

The PARSEVAL measures have been criticized for being too permissive in some situations while sometimes penalizing the same error more than once (Lin, 1995; Carroll and Briscoe, 1996; Carpenter and Manning, 1997; Carroll and Briscoe, 1996). Regardless of these problems, however, the PARSEVAL measures and the attachment scores for dependency representations have the disadvantage that they are only applicable to one kind of representation. As an alternative to these metrics, several researchers have therefore proposed evaluation schemes based on dependency structure, where both the treebank annotation and the parser output, whether constituency-based or dependency-based, are converted into sets of more abstract dependency relationships (Lin, 1995, 1998; Carroll et al., 1998; Kübler and Telljohann, 2002; Carroll et al., 2003). Recently, this has led to the development of *dependency banks* for parser evaluation (Carroll et al., 2003; King et al., 2003; Forst et al., 2004). Having more abstract representations also makes the scheme less sensitive to the indeterminacy problem in annotation. The only drawback with this methodology is the overhead involved in converting parser representations to the more abstract dependency relationships and the possible errors that may be introduced in this process. In situations where only one kind of representation is relevant, it may therefore still be justified to use the more representation-dependent metrics. Thus, in the experiments reported below we will mainly use metrics based on exact match and attachment scores for dependency-based representations.

5.1.2 Treebank Learning

In the data-driven approach to text parsing, treebank data is crucial not only for the evaluation but also for the development of parsing systems, since the core component of this approach is the application of inductive inference to a representative sample of data in the training phase. In principle, the use of treebank data in the training phase is independent of its use in evaluation, but in practice they are intimately connected since the development of a data-driven parser normally involves an iterative training-evaluation cycle, where different parameters are varied systematically to improve overall performance.

During both development and final evaluation, it is essential that the data used for evaluation is distinct from the data used for training. In both cases, we are interested in estimating the expected accuracy, i.e., the accuracy that we can expect on average when applying the parser to an independent test set, and training set accuracy is in general a very poor estimate of this quantity. Training set accuracy increases consistently with model complexity, but a model with very high training set accuracy often overfits the training data and does not generalize well (Hastie et al., 2001).

However, repeatedly using the same test set for evaluation will produce a similar effect, which means that the test set accuracy may substantially overestimate the expected accuracy on unseen data. It is important in this context to distinguish two different but related problems: *model selection* and *model assessment* (Hastie et al., 2001). Model selection is the problem of estimating the performance of different models in order to choose the (approximate) best one; model assessment is the problem of estimating the expected accuracy of the finally selected model.

In a data-rich situation, the standard solution is to randomly divide the available data into three parts: a training set, a validation set, and a test set. The training set is used for inductive inference; the validation set is used (repeatedly) to estimate accuracy for model selection; and the test set is used for the assessment of the accuracy of the final chosen model. A well-known example of this methodology is the standard split of the Wall Street Journal section of the Penn Treebank into sections 02–21 for training, one of the sections 00, 22 and 24 for validation, and section 23 for final testing.

In a data-poor situation, there are various techniques that can be used to approximate the validation step without having a separate validation set, either by analytical methods, such as Bayesian Information Criterion (BIC) or Minimum Description Length (MDL), or by efficient sample re-use, such as cross-validation and bootstrap methods (Hastie et al., 2001). Although these techniques can also to some extent be used for model assessment, it is more common to combine them with an independent test set for the final evaluation.

Whether treebank parsing should be considered a data-rich or a data-poor situation depends to some extent on which languages we are interested in. For English, there are several treebanks of reasonable size available, which explains the standard training-validation-test setup usually applied to the Wall Street

Journal data. However, it is also worth pointing out that, even if section 23 is only used once in every published study based on this data set, it has over the years been used repeatedly by the same and different research groups, which in fact amounts to a kind of repeated testing, albeit at a higher level of abstraction. Thus, the value of this data set as a basis for estimation of expected accuracy is by now highly dubious, and it is probably better regarded today as a benchmark set.

For a language like Swedish, which is of special interest in this study, the availability of treebank data is rather limited, which motivates the use of cross-validation for model selection, reserving a separate test set for the final evaluation. Finally, it is worth remembering that for most languages of the world, there are simply no treebank data available at all, which rules out supervised learning methods completely.

5.1.3 Treebanks for Dependency Parsing

When the data-driven approach to text parsing is combined with supervised learning methods, training data must be annotated with the same kind of syntactic representations that are used in the parsing system. In our case, this means that we require treebanks that are annotated with dependency graphs. The availability of such treebanks has increased substantially in recent years. In addition to the Prague Dependency Treebank of Czech (Hajič, 1998; Hajič et al., 2001), which is probably the most well-known treebank of this kind, we find the METU Treebank of Turkish (Oflazer et al., 2003), the Danish Dependency Treebank (Kromann, 2003), the Eus3LB Corpus of Basque (Aduriz et al., 2003), the Turin University Treebank of Italian (Bosco and Lombardo, 2004), and the parsed corpus of Japanese described in Kurohashi and Nagao (2003). Furthermore, there are hybrid treebanks, which include both constituency and dependency annotation, such as the TIGER Treebank of German (Brants et al., 2002) and the Alpino Treebank of Dutch (Van der Beek et al., 2002).

In fact, whereas many of the early large-scale treebank projects, such as the Lancaster Parsed Corpus (Garside et al., 1992) and the original Penn Treebank (Marcus et al., 1993), were based on constituency annotation only, most annotation schemes today include some kind of functional analysis that can be regarded as a partial dependency analysis. This is true of the Penn Treebank II annotation scheme (Bies et al., 1995), which adds functional tags to the original phrase structure annotation, and similar combinations of constituent structure and grammatical functions are found in the SUSANNE annotation scheme (Sampson, 1995), in the ICE-GB Corpus of British English (Nelson et al., 2002), and in the adaptations of the Penn Treebank II schemes that have been developed for Chinese (Xue et al., 2004), Korean (Han et al., 2002), Arabic (Maamouri and Bies, 2004) and Spanish (Moreno et al., 2003).

Constituency-based treebanks, with or without functional annotation, can in principle be converted to dependency treebanks. As shown by Gaifman (1965), it is straightforward to convert a constituency tree to an unlabeled

dependency tree, provided that every constituent c has a unique head child c_h. The dependency tree is obtained by recursively letting the head d of each non-head child c_d of c be a dependent to the head h of the head child c_h of c (where a terminal node $c_h = h$ is its own head) (cf. Xia and Palmer, 2001). This method has been used in several studies to convert constituency-based treebank annotations to dependency structures, notably using data from the Penn Treebank (Collins, 1996, 1997, 1999; Xia and Palmer, 2001; Xia, 2001; Yamada and Matsumoto, 2003; Nivre and Scholz, 2004), but also from the German treebanks TüBa-D (Kübler and Telljohann, 2002) and TIGER (Bohnet, 2003; Ule and Kübler, 2004).

In practice, there are normally a number of factors that interact to make some of the converted dependency representations less than optimal. Besides problems that are inherent in the dependency-based approach to syntactic representations, such as the existence of constructions that are not readily analyzed as single-headed, the main problem is that it may be difficult to identify the head child in a constituency representation even when such a child exists, since most constituency-based annotation schemes do not mark heads explicitly. The standard solution to this problem is to use *head percolation tables* (Magerman, 1995; Collins, 1996, 1999) that provide heuristic rules for identifying the head child in a constituent of a specific type. Figure 5.1 shows the head percolation table used by Yamada and Matsumoto (2003) and Nivre and Scholz (2004). The first column contains the constituent labels found in the Penn Treebank. For each constituent label, the second column specifies the direction of search (from the right [R] or from the left [L]) and the second column gives a list of potential head child categories, partially ordered by descending priority (with the vertical bar | symbolizing equal priority). For example, for a constituent of type NP, we start searching from the right for a child of type POS, NN, NNP, NNPS or NNS. The first child matching this condition is chosen as the head. If no child matching this condition is found, we proceed to search for a child of type NX, etc. If the entire list is exhausted, the first child encountered when searching in the specified direction is chosen as the head.

When applied to the Penn Treebank, a head percolation table of this kind gives a quite reasonable conversion to dependency structures for the majority of constituent types. However, for certain types of constituents, such as complex noun phrases involving coordination, it is extremely difficult to devise a set of rules that guarantees an adequate conversion in all cases. The most elaborate scheme in this respect is probably the rules used by Collins (1999), where the head percolation table is supplemented by special rules for noun phrases and coordination (cf. also Bikel, 2004).

The problem of identifying head children in constituency representations is mitigated if an extensive functional annotation is present. Thus, in converting the Swedish treebank Talbanken (Einarsson, 1976a,b) to a dependency treebank, the problem of identifying head children can be solved almost completely by only considering the functional annotation (Nilsson et al., 2005).

NP	R	POS|NN|NNP|NNPS|NNS NX JJR CD JJ JJS RB QP NP
ADJP	R	NNS QP NN $ ADVP JJ VBN VBG ADJP JJR NP JJS DT FW RBR RBS SBAR RB
ADVP	L	RB RBR RBS FW ADVP TO CD JJR JJ IN NP JJS NN
CONJP	L	CC RB IN
FRAG	L	
INTJ	R	
LST	L	LS :
NAC	R	NN|NNS|NNP|NNPS NP NAC EX $ CD QP PRP VBG JJ JJS JJR ADJP FW
PP	L	IN TO VBG VBN RP FW
PRN	R	
PRT	L	RP
QP	R	$ IN NNS NN JJ RB DT CD NCD QP JJR JJS
RRC	L	VP NP ADVP ADJP PP
S	R	TO IN VP S SBAR ADJP UCP NP
SBAR	R	WHNP WHPP WHADVP WHADJP IN DT S SQ SINV SBAR FRAG
SBARQ	R	SQ S SINV SBARQ FRAG
SINV	R	VBZ VBD VBP VB MD VP S SINV ADJP NP
SQ	R	VBZ VBD VBP VB MD VP SQ
UCP	L	
VP	L	VBD VBN MD VBZ VB VBG VBP VP ADJP NN NNS NP
WHADJP	R	CC WRB JJ ADJP
WHADVP	L	CC WRB
WHNP	R	WDT WP WP$ WHADJP WHPP WHNP
WHPP	L	IN TO FW
NX	R	POS|NN|NNP|NNPS|NNS NX JJR CD JJ JJS RB QP NP
X	R	

Fig. 5.1. Head percolation table for the Penn Treebank

However, the presence of functional annotation is even more crucial if we want to convert constituent representations to *labeled* dependency graphs, since dependency type labels can normally be inferred from the functional annotation but only indirectly from the constituency annotation.

The experiments presented in this chapter are based on data from Swedish and English. The choice of these languages does not reflect the availability of dependency treebanks but rather a desire to develop better parsing systems for Swedish, on the one hand, and to compare the performance of the system to available benchmarks for English, on the other hand. An unfortunate consequence of this choice is that the experiments will in both cases be performed on dependency treebanks that are the result of conversion from another kind of annotation. Experiments on genuine dependency treebanks, notably the Prague Dependency Treebank and the Danish Dependency Treebank, have also been performed but will not be reported here. The main reason is that

Genom	PR	AAPR
skattereformen	NNDDSS	AA
införs	VVPSSMPA	FV
individuell	AJ	SSAT
beskattning	VN	SS
av	PR	SSETPR
arbetsinkomster	NN SS	SSET
.	IP	IP

Fig. 5.2. Swedish sentence annotated according to MAMBA

this requires special treatment of non-projective dependency graphs, which would take us too far afield in this study. Preliminary results on the use of inductive dependency parsing in combination with graph transformation techniques to capture non-projective structures are reported by Nivre and Nilsson (2005).

5.2 Experimental Methodology

As noted above, the experimental evaluation of the deterministic and memory-based version of inductive dependency parsing is based on data from two languages, Swedish and English. We will systematically vary parameters of the feature model and the learning algorithm in order to study their influence on parsing accuracy and efficiency. We will also compare the results to relevant previous research. In this methodological section, we first describe the data sets used in the experiments, the parameters varied in the experiments, and the metrics used to evaluate parsing accuracy and efficiency.

5.2.1 Treebank Data

The Swedish data for the experiments come from Talbanken (Einarsson, 1976a,b), a syntactically annotated corpus of written and spoken Swedish, created in a series of projects at the University of Lund in the 1970s. In this study, we use the Professional Prose section, containing informative and argumentative text from brochures, newspapers, and books. The syntactic annotation follows the MAMBA scheme (Teleman, 1974), which is described by its creators as an eclectic combination of constituent structure, dependency structure and topological field analysis. Figure 5.2 shows an example of the MAMBA annotation applied to a Swedish sentence taken from Talbanken.[1]

[1] Word-by-word gloss: 'Through tax-reform-DEF introduce-PAST-PASSIVE individual taxation of work-income-PLUR.' Translation: 'Through the tax reform individual taxation of work incomes is introduced.'

The annotation consists of two layers, the first being a lexical analysis, consisting of part-of-speech information including morphological features, and the second being a syntactic analysis, in terms of grammatical functions. Both layers are flat in the sense that they consist of tags assigned to individual word tokens, but the syntactic layer also gives information about constituent structure, as exemplified with respect to the grammatical subject in figure 5.2. All the words belonging to the subject noun phrase *individuell beskattning av arbetsinkomster* (individual taxation of work incomes) are annotated with the tag SS for subject, but the head noun *beskattning* (taxation) is marked as such by having only this tag, while the pre-modifying adjective *individuell* (individual) is also tagged AT for adjectival modifier and the words of the post-modifying prepositional phrase *av arbetsinkomster* (of work incomes) are tagged ET for nominal post-modifier. Within the prepositional phrase, the noun *arbetsinkomster* (work incomes) is marked as the head, while the preposition *av* (of) gets an additional tag PR for the prepositional function.

The constituent structure recognized in MAMBA is rather flat, especially on the clause level where the analysis to a large extent is modeled after the topological field analysis proposed by Diderichsen (1946) for the Scandinavian languages. The main constituents recognized in the clause are the following:

- Verb (–V)
- Subject (–S)
- Object (–O)
- Predicative (–P)
- Adverbial (–A)

A more fine-grained classification of these constituents is obtained by varying the first letter of the two-letter tag. Thus, finite verbs are tagged FV, non-finite verbs IV; logical subjects are tagged ES, formal subjects FS, and other subjects SS, etc. In addition to the constituents recognized in Diderichsen's topological field model, there is a limited phrase structure analysis of noun phrases, prepositional phrases, adjective phrases, and subordinate clauses, with special tags for internal grammatical functions. Altogether, there are 42 distinct grammatical function tags in the MAMBA annotation scheme (Teleman, 1974).

Thanks to the rich functional annotation it is relatively straightforward to convert the MAMBA annotation to dependency graphs. For the majority of phrases, the syntactic head is explicitly marked and the function tags assigned to other constituents can be used to label dependency arcs. However, there are two types of structures that require special treatment. The first is the clause structure, where there is no explicit indication of a syntactic head, and where the following categories are considered as candidate heads in descending order of priority:

1. The leftmost finite verb (FV).
2. The leftmost non-finite verb (IV).

ADV	Adverbial modifier
APP	Apposition
ATT	Attribute (adnominal modifier)
CC	Coordination (conjunction or second conjunct)
DET	Determiner
ID	Non-first element of multi-word expression (idiom)
IM	Infinitive dependent on infinitive marker
INF	Infinitival complement
IP	Punctuation
OBJ	Object
PR	Complement of preposition
PRD	Predicative complement
ROOT	Dependent of special root node
SUB	Subject
UK	Head verb of subordinate clause dependent on complementizer
VC	Verb chain (nonfinite verb dependent on other verb)
XX	Unclassifiable dependent

Fig. 5.3. Dependency types in Swedish treebank

3. The head of the leftmost predicative complement (–P).
4. The head of the leftmost subject, object or adverbial (–S, –O, –A).
5. The leftmost word.

The second type of structure is coordination, where the MAMBA annotation treats every conjunct as a head. In our conversion to dependency structure, we adopt a Mel'čuk style analysis of coordination and treat the leftmost conjunct as the head (cf. section 3.1.2).

After the initial conversion to dependency graphs, we apply two types of transformations to these graphs. The first is related to some of the traditional open issues in dependency grammar, where we prefer a different analysis than the one assumed in the MAMBA annotation. More precisely:

1. Prepositional phrases are headed by the preposition, which takes the head noun of the nominal complement as a dependent.
2. Subordinate clauses with an overt complementizer (except relative clauses) are headed by the complementizer, which takes the finite verb of the subordinate clause as a dependent.
3. Infinitival verb phrases with an overt infinitive marker are headed by the infinitive marker, which takes the infinitive verb as a dependent.

The second type of transformation concerns three kinds of structure that do not have a clear-cut dependency analysis, namely idioms (including multiword proper names and compound function words), verb chains and coordinate structures. By analogy with Tesnière's notion of dissociate nuclei, these

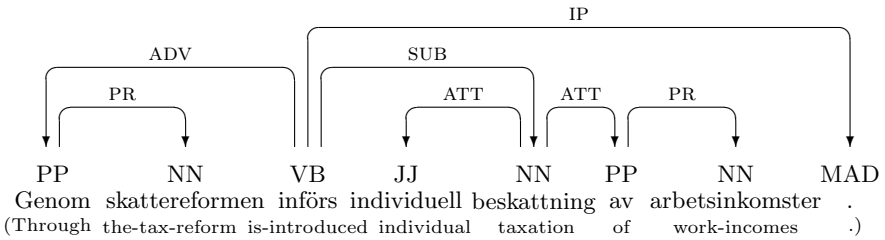

Fig. 5.4. Dependency graph for Swedish sentence, converted from Talbanken

constructions are treated as left-headed chains, where each subsequent element is a dependent of the immediately preceding one, and where left and right dependents of the entire unit are attached to the leftmost and rightmost element, respectively (and internal dependents to the leftmost element). In the case of verb chains, this means that left dependents will formally be treated as dependents of the leftmost verb (normally the finite verb in Swedish), while right dependents will be attached to the rightmost verb (possibly a non-finite verb).

The final thing to note about the conversion of the MAMBA annotation is the set R of dependency types used as arc labels. On the one hand, we have collapsed some of the finer distinctions in the original set of grammatical functions, where no less than twelve different types of adverbials are distinguished. On the other hand, we have added a few dependency types for relations that are not marked explicitly in the MAMBA annotation, notably for verb chains and coordination. This gives us a set R of 17 dependency labels (including the special root label r_0 = ROOT), which are listed with explanations in figure 5.3.

The part-of-speech tags included in the original annotation of Talbanken have not been used in the experiments, mainly because there is no part-of-speech tagger available for this tagset. Since we want to be able to apply the parsing system to new texts, we have therefore used a statistical tagger trained on the much larger Stockholm-Umeå Corpus (Ejerhed and Källgren, 1997), using a tagset consisting of 150 tags, to preprocess the Swedish data both for training and for evaluation. The estimated accuracy of the tagger, when evaluated on held-out data from the Stockholm-Umeå Corpus is 94.4%. Figure 5.4 shows the result of converting the sentence in figure 5.2 using the procedure described in this section and tagging it with the statistical part-of-speech tagger (although the morphological features of the part-of-speech tags have been suppressed for readability reasons).

The dependency treebank obtained by converting the Professional Prose section of Talbanken consists of 6316 sentences and 97623 tokens (including punctuation), which gives a mean sentence length of 15.46 tokens. For previous experiments, the sentences of this treebank have been randomly divided into

```
( (S
     (NP-SBJ (JJ Economic) (NN news) )
     (VP (VBD had)
       (NP
         (NP (JJ little) (NN effect) )
         (PP (IN on)
           (NP (JJ financial) (NNS markets) ))))
     (. .) ))
```

Fig. 5.5. English sentence annotated according to Penn Treebank II

ten equally large sections, numbered 0–9, where sections 1–8 have been used as training data and section 9 as validation data, saving section 0 for later studies (Nivre et al., 2004; Nivre and Nilsson, 2004; Nivre, 2004a). In this study, we instead use nine-fold cross-validation on sections 1–9 for model selection, and use section 0 as the test set for the final model assessment. The data set used for cross-validation consists of 5685 sentences and 87757 tokens, while the final test set consists of 631 sentences and 9841 tokens.

The English data are taken from the Penn Treebank (Marcus et al., 1993), which has been the most widely used treebank for parser evaluation over the last decade. In this study, we use the Wall Street Journal section of the treebank, with the Penn Treebank II annotation scheme (Bies et al., 1995), which combines constituency analysis with a limited functional annotation. Figure 5.5 repeats the example sentence used in chapters 1–4, this time in the original annotation format using the full node labels, composed of bracketing labels and grammatical function labels.

We assume that the Penn Treebank II annotation scheme is familiar to most readers and proceed directly to a discussion of the way in which this annotation can be converted to dependency graphs. For the unlabeled dependency graphs we rely on the standard method described in section 5.1.3, using the head percolation table of Yamada and Matsumoto (2003), which is a slight modification of the rules used by Collins (1999). Using this conversion scheme permits us to make exact comparisons with the parser of Yamada and Matsumoto (2003), as well as the parsers of Collins (1997) and Charniak (2000), which are evaluated on the same data set in Yamada and Matsumoto (2003). The head percolation table can be found in figure 5.1.

In addition to the structural conversion, we also have to derive dependency types to use as arc labels. Compared to the Swedish treebank, the functional annotation in the Penn Treebank is much less comprehensive, which makes this a non-trivial problem. Given an arc $i \rightarrow j$, derived from a local constituent tree where w_i is the head of the head child h and w_j is the head of a non-head child d, let M, H and D be the original labels on the mother node, head child h and child d, respectively, except that H and D are replaced by

AMOD	Modifier of adjective or adverb (phrase adverbial)
DEP	Other dependent (default label)
NMOD	Modifier of noun (including complement)
OBJ	Object
P	Punctuation
PMOD	Modifier of preposition (including complement)
PRD	Predicative complement
ROOT	Dependent of special root node
SBAR	Head verb of subordinate clause dependent on complementizer
SBJ	Subject
VC	Verb chain (nonfinite verb dependent on other verb)
VMOD	Modifier of verb (sentence or verb phrase adverbial)

Fig. 5.6. Dependency types in English treebank

TAG if they are part-of-speech tags. The labels M, H and D, stripped of their function tags, have been used by Collins (1999) to construct complex dependency labels (M, H, D, dir), where dir is L or R (for left and right dependency, respectively). In our experiments, we instead use these labels to formulate a set of rules for choosing the arc label r, $i \xrightarrow{r} j$. In order of descending priority, the rules are as follows:

1. If D is a punctuation category, r = P.
2. If D contains the function tag SBJ, r = SBJ.
3. If D contains the function tag PRD, r = PRD.
4. If M = VP, H = TAG and D = NP (without any function tag), r = OBJ.
5. If M = VP, H = TAG and D = VP, r = VC.
6. If M = SBAR and D = S, r = SBAR.
7. If M = VP, S, SQ, SINV or SBAR, r = VMOD.
8. If M = NP, NAC, NX or WHNP, r = NMOD.
9. If M = ADJP, ADVP, QP, WHADJP or WHADVP, r = AMOD.
10. If M = PP or WHPP, r = PMOD.
11. Otherwise, r = DEP.

The complete set R of dependency types, including the root label r_0 = ROOT, is listed in figure 5.6. The explanations given reflect the intended interpretation of each category, although it is clear that the rules for choosing dependency types will also cover cases that do not fit the descriptions. A notoriously difficult problem is the treatment of complex noun phrases, which have a very flat structure in the Penn Treebank. For example, coordinated noun phrases will often be analyzed as structures where the last conjunct is the head, while preceding conjuncts as well as the coordinating conjunction are analyzed as dependents of the NMOD type. Similar problems can be identified for most of the rules and categories. However, having a set of dependency types that are

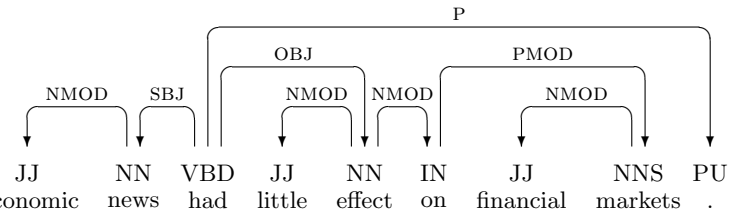

Fig. 5.7. Dependency graph for English sentence, converted from the Penn Treebank (cf. figures 1.1, 2.2 and 3.1)

Table 5.1. Data sets for training, validation and test; Sec: section, W: number of tokens, S: number of sentences, W/S: mean number of tokens per sentence

	Swedish				**English**			
Data set	**Sec**	**W**	**S**	**W/S**	**Sec**	**W**	**S**	**W/S**
Training	1–9	87757	5685	15.44	01-21	950028	39832	23.85
Validation	–	–	–	–	00	46451	1921	24.18
Test	0	9841	631	15.60	23	56684	2416	23.46

similar in nature and cardinality to the Swedish set will make results more comparable.

As regards part-of-speech tagging, we use the gold standard tags from the Penn Treebank for training, and a statistical tagger trained on section 2-21 for validation and final testing. The tagger has an accuracy of 96.1% on section 23. In the final evaluation, we will also evaluate the parser on gold standard tags to see how large proportion of the errors can be attributed to tagging errors. Figure 5.7 shows the result of converting the sentence in figure 5.5 using the procedure described in this section (with the gold standard tags from the treebank).

The dependency treebank obtained by converting the Wall Street Journal section of the Penn Treebank consists of 49208 sentences and 1173766 tokens, which gives a mean sentence length of 23.85 words. We use sections 2-21 for training (39832 sentences, 950028 tokens), section 00 for validation and model selection (1921 sentences, 46451 tokens), and section 23 for the final model assessment (2416 sentences, 56684 tokens).

Table 5.1 gives an overview of all the data sets used in the experiments. There is no separate validation data set for Swedish, given that we use cross-validation for model selection.

5.2.2 Models and Algorithms

As emphasized on several occasions, the parsing methods evaluated in these experiments are only one possible instantiation of the general framework of inductive dependency parsing. This means that, although the experiments will involve a systematic study of the influence of different variables on accuracy and efficiency, there are also many factors that will be kept constant. This holds in particular for the parsing algorithm, which will in all cases be the deterministic arc-eager algorithm presented and analyzed in section 3.4. But it also holds for the learning method, in the sense that we will only consider memory-based learning algorithms, as described in section 4.3, although we will explore many variants of this general approach to machine learning.

The first part of the experiment, presented in section 5.3, will be devoted to parameters of the feature model, exploring different combinations of the three types of features discussed in section 4.2: part-of-speech features, dependency features, and lexical features. We will begin with simple models based only on part-of-speech features and gradually increase model complexity by adding first dependency features and then lexical features. The different models will mainly be evaluated with respect to parsing accuracy, but a selected subset will also be evaluated for efficiency. Finally, we will consider the learning curves of different models, i.e., parsing accuracy as a function of the size of the training corpus. Throughout the first part, the parameters of the learning algorithm will be kept constant. As described in section 4.3.2, we will use the following settings for the k-NN classification provided by the memory-based learner:

1. Number of nearest distances: $k = 5$
2. Distance metric: MVDM with $l = 3$
3. Feature weighting: None
4. Distance-weighted class voting: ID weighting.

In terms of the TiMBL system, this corresponds to the following parameter settings: -k 5 -m M -L 3 -w 0 -d ID (cf. Daelemans et al., 2004).

The second part of the experiment, presented in section 5.4, will explore some of the options provided by TiMBL for the memory-based learning and classification. In particular, we will consider the influence of different k values and distance metrics, in interaction with different schemes for feature weighting and distance-weighted voting. Throughout the second part, the parameters of the feature model will in principle be kept constant, but we will consider two different feature models, one that is lexicalized and one that is not.

The first two parts of the experiment constitute the validation or model selection phase, in the terminology of section 5.1.2. The third and final part of the experiment is the final evaluation or model assessment phase, where we apply the best models with the best settings to a test data set that has not been used in the validation phase. However, it is important to keep in mind that, because of the complex interaction of feature models, learning

algorithm parameters and properties of the data sets, there is no guarantee that the models and settings selected for the final evaluation are in fact truly optimal even for the given data sets.

5.2.3 Evaluation

We will use two different metrics to evaluate parsing accuracy, *attachment score* (AS) and *exact match* (EM). AS measures the proportion of *tokens* that are correctly analyzed, while EM measures the proportion of *sentences* that are assigned a completely correct dependency graph. Both metrics come in an unlabeled version, which only considers the attachment of dependents to head, and a labeled version, which also takes the dependency type labels into account.

Definition 5.1. Given a test sample $T_e = (x_1, \ldots, x_m)$, consisting of m sentences, where each sentence $x_i = (w_1, \ldots, w_k)$ consists of k_i tokens, and the total number of tokens in the sample is n (i.e., $n = \sum_{i=1}^{m} k_i$). Let $A_g = (G_1^g, \ldots, G_m^g)$ be the dependency graphs of the gold standard annotation, let $A_p = (G_1^p, \ldots, G_m^p)$ be the dependency graphs produced by parser p, let $G_i^g = (V_{x_i}, E_i^g, L_i^g)$ and $G_i^p = (V_{x_i}, E_i^p, L_i^p)$, and let h_i^g, d_i^g, h_i^p and d_i^p be the following functions:

$$\begin{aligned} h_i^g(j) = l &\;\Leftrightarrow\; (l, j) \in E_i^g \\ d_i^g(j) = r &\;\Leftrightarrow\; \exists l : ((l, j), r) \in L_i^g \\ h_i^p(j) = l &\;\Leftrightarrow\; (l, j) \in E_i^p \\ d_i^p(j) = r &\;\Leftrightarrow\; \exists l : ((l, j), r) \in L_i^p \end{aligned}$$

The *unlabeled attachment score* AS_U of p with respect to T_e and A_g is:

$$\mathrm{AS}_U = \frac{1}{n} \sum_{i=1}^{m} \sum_{j=1}^{k_i} \delta(h_i^g(j), h_i^p(j))$$

The *labeled attachment score* AS_L of p with respect to T_e and A_g is:

$$\mathrm{AS}_L = \frac{1}{n} \sum_{i=1}^{m} \sum_{j=1}^{k_i} \delta(h_i^g(j), h_i^p(j)) \cdot \delta(d_i^g(j), d_i^p(j))$$

The *unlabeled exact match* EM_U of p with respect to T_e and A_g is:

$$\mathrm{EM}_U = \frac{1}{m} \sum_{i=1}^{m} \delta(h_i^g, h_i^p)$$

The *labeled exact match* EM_L of p with respect to T_e and A_g is:

$$\mathrm{EM}_L = \frac{1}{m} \sum_{i=1}^{m} \delta(h_i^g, h_i^p) \cdot \delta(d_i^g, d_i^p)$$

In the final evaluation in section 5.5, we will also provide a breakdown of the attachment score for different dependency types, which will be computed as *unlabeled attachment score*, (labeled) *precision*, (labeled) *recall* and (labeled) F_β *measure* ($\beta = 1$) for each dependency type r.

Definition 5.2. Let T_e be a test sample and let h_i^g, d_i^g, h_i^p and d_i^p be defined as in definition 5.1. The *unlabeled attachment score* $AS_U(r)$ of dependency type r for parser p with respect to T_e and A_g is:

$$\mathrm{AS}_U(r) = \frac{|\{w_j \in T_e \mid d_i^g(j) = r, h_i^g(j) = h_i^p(j)\}|}{|\{w_j \in T_e \mid d_i^g(j) = r\}|}$$

The *precision* $\mathrm{P}(r)$ of r for p with respect to T_e and A_g is:

$$\mathrm{P}(r) = \frac{|\{w_j \in T_e \mid d_i^p(j) = d_i^g(j) = r, h_i^g(j) = h_i^p(j)\}|}{|\{w_j \in T_e \mid d_i^p(j) = r\}|}$$

The *recall* $\mathrm{R}(r)$ of r for p with respect to T_e and A_g is:

$$\mathrm{R}(r) = \frac{|\{w_j \in T_e \mid d_i^p(j) = d_i^g(j) = r, h_i^g(j) = h_i^p(j)\}|}{|\{w_j \in T_e \mid d_i^g(j) = r\}|}$$

The F *measure* $\mathrm{F}(r)$ of r for p with respect to T_e and A_g is:

$$\mathrm{F}(r) = \frac{2 \cdot \mathrm{P}(r) \cdot \mathrm{R}(r)}{\mathrm{P}(r) + \mathrm{R}(r)}$$

While the $\mathrm{AS}_U(r)$ score only measures how often a dependent of type r is assigned the correct head (regardless of the assigned label), the $\mathrm{P}(r)$ score tells us how often the parser is completely correct when using the label r and the $\mathrm{R}(r)$ score how often a dependent of type r is parsed completely correctly, while $\mathrm{F}(r)$ is the harmonic mean of $\mathrm{P}(r)$ and $\mathrm{R}(r)$.

In all scores reported for evaluation metrics concerning accuracy, punctuation tokens will be omitted from the counts. Figure 5.8 lists the parts-of-speech that are counted as punctuation categories in the two treebanks.

Efficiency will be evaluated by the following three metrics:

1. Training time: The time required to construct the instance base for the memory-based classifier, including the parsing of the training corpus using the gold standard parsing algorithm and the precomputation of metrics by TiMBL.
2. Parsing time: The time required to parse one sentence of the test corpus, excluding the initialization of the parser.
3. Memory consumption: The amount of memory allocated for parsing the test corpus.

Although parsing time is measured for each individual sentence, the results will mostly be presented in aggregated form, as the total parsing time for

Swedish		English	
MAD	Major delimiter	.	Sentence-final punctuation
MID	Minor delimiter	,	Comma
		:	Colon, semi-colon
PAD	Paired delimiter	-LRB-	Left bracket character
		-RRB-	Right bracket character
		"	Straight double quote
		‘	Left open single quote
		“	Left open double quote
		’	Right close single quote
		”	Right close double quote
		#	Pound sign
		$	Dollar sign

Fig. 5.8. Punctuation categories in Swedish and English treebank

a given test corpus, as the mean parsing time per sentence, or as the mean number of words parsed per second (cf. table 5.1 for quantitative properties of the data sets). Time is measured using standard system calls, with measurements reported in seconds (s) or milliseconds (ms), while memory consumption is measured through the UNIX command `top` and reported in number of megabytes (MB). All experiments are run on a SunBlade 2000 with one 1.2GHz UltraSPARC-III processor and 1GB of memory.

The results presented in section 5.3–5.4 are based on the cycle of training and validation for model selection. For the smaller Swedish data set, validation is performed by means of nine-fold cross-validation on sections 1–9, and all results presented are the arithmetic mean of the results from the nine folds. For the larger English data set, we consistently use sections 02–21 for training and section 00 for validation. The results presented in section 5.5 are the final results for model assessment, which involve training on sections 1–9 and testing on section 0 for Swedish, training on sections 02-21 and testing on section 23 for English. For the final evaluation, we use McNemar's test to assess the statistical significance of differences in accuracy (both attachment score and exact match).

5.3 Feature Model Parameters

We start our investigation of different feature models from a baseline model, where the only features used to predict the next transition are the parts-of-speech of the top token and the next token. In the notation introduced in section 4.2.4, the baseline is written Φ^p_{00}, and its two features $p(\sigma_0)$ and $p(\tau_0)$. In the next three sections, we will gradually increase the complexity of the model by adding a larger part-of-speech context, dependency features, and

Table 5.2. Accuracy as a function of part-of-speech context only; AS: attachment score, EM: exact match; U: unlabeled, L: labeled; Swedish (cross-validation), English (section 00)

Model	Swedish				English			
	AS		EM		AS		EM	
	U	L	U	L	U	L	U	L
Φ^p_{00}	71.0	64.5	15.6	12.3	58.4	56.2	3.7	3.1
Φ^p_{01}	73.8	67.5	17.8	13.8	75.9	73.1	8.5	6.8
Φ^p_{10}	74.8	67.8	23.4	15.5	60.5	58.0	5.0	4.0
Φ^p_{11}	**77.9**	**70.9**	**27.2**	**18.0**	**77.7**	**74.8**	**13.4**	**10.2**
Φ^p_{21}	**78.4**	**71.0**	**28.4**	**18.6**	**77.7**	**74.8**	**14.2**	**10.7**
Φ^p_{31}	78.0	70.0	28.1	18.4	77.1	74.2	13.6	10.0
Φ^p_{12}	**77.4**	**70.1**	**26.6**	**17.8**	**79.0**	**76.1**	14.4	10.0
Φ^p_{13}	77.1	69.7	26.1	17.3	78.8	75.9	13.6	9.5
Φ^p_{14}	77.1	69.7	26.3	17.3	78.8	75.9	**14.6**	**10.2**
Φ^p_{15}	77.0	69.7	26.0	17.3	78.5	75.5	14.1	9.6
Φ^p_{16}	77.0	69.6	25.9	17.2	78.7	75.7	14.3	9.3

lexical features. Finally, we will evaluate a subset of the models with respect to efficiency and also investigate their learning curves.

5.3.1 Part-of-Speech Context

The role of part-of-speech features in data-driven approaches to parsing is far from clear-cut, as noted in section 4.2.2, but they are nevertheless used in most models and appear to have a positive effect especially by providing a backoff model for lexical features (Charniak, 2000; Van den Bosch and Buchholz, 2002).

Table 5.2 shows the accuracy obtained for Swedish and English with feature models that differ only with respect to the number of tokens included in the part-of-speech context. Remember that Φ^p_{mn} is a model that includes the $m+1$ top tokens on the stack and the $n+1$ next input tokens (cf. section 4.2.4).

Our first observation is that the baseline model achieves a very modest parsing accuracy, both with respect to attachment score and exact match, especially for English. We see that adding a lookahead of just one token (Φ^p_{01}) makes a tremendous difference for English, and is clearly beneficial for Swedish as well. We see that adding one more token from the stack (Φ^p_{10}) also has a positive effect, although in this case the difference is greater for Swedish, where it has an especially strong effect on the exact match evaluation. For English, the strong positive effect on the exact match evaluation shows up only when the extra stack token is combined with an extra input token (Φ^p_{11}).

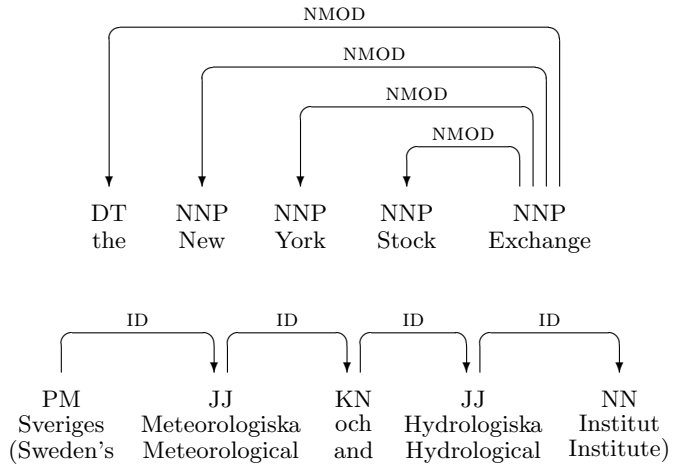

Fig. 5.9. Nominal compounds and multi-word names in English and Swedish

Considering the results at a superficial level, it seems that looking forward is more important for English, while looking backwards is more important for Swedish. However, this difference can to a very large extent be explained by the proliferation of nominal compounds in English, in combination with the particular dependency analysis inherited from the converted phrase structure annotation and the deterministic parsing strategy used. To illustrate this phenomenon, let us consider a typical example from the Wall Street Journal data:

the New York Stock Exchange (5.4)

Apart from the determiner *the*, this noun phrase consists of four consecutive words, which in the Penn Treebank annotation are all tagged as proper nouns (NNP). According to the head percolation table, a noun phrase of this kind is always headed by the rightmost noun, in this case the noun *Exchange*, with all the preceding words as dependents. In order to parse such a structure correctly, the parser must keep shifting until the head noun is the next input token and then perform a series of LEFT-ARC(NMOD) transitions until all the dependents have been attached to the noun. However, without any form of lookahead it will be very hard to predict when the last noun has been encountered, especially with a feature model that only includes part-of-speech features.

The same problem does not arise with the Swedish data. First of all, Swedish compounds are normally written as single words, which means that nominal compounds will not be encountered as syntactic units in the Swedish data. Moreover, in structures similar to the English example, such as multi-word proper names, the Swedish annotation marks the leftmost element as the

head and treats each subsequent element as a dependent of the immediately preceding element. Therefore, the problem of predicting when the head has been found does not occur when parsing Swedish. The difference between the two styles of analysis are illustrated in figure 5.9, which contrast the annotation of English nominal compounds in the converted Penn Treebank with the annotation of Swedish multi-word units in Talbanken.

Moving on to the middle section of table 5.2, we see that adding a second extra stack token (Φ^p_{21}) gives a marginal increase in accuracy, especially with respect to exact match, but that adding a third token (Φ^p_{31}) gives a decrease across the board. In the lower section, we see essentially the same pattern with respect to lookahead, although with a differentiation between languages. For English, adding a second lookahead token (Φ^p_{12}) is beneficial but extending the context even further is not. For Swedish, accuracy starts to go down already with the second lookahead token. However, we will see later that the effect of increasing the part-of-speech context is also sensitive to the presence of other features.

Summarizing the results for part-of-speech features only, it seems that the model Φ^p_{21} gives the best performance for Swedish, while the model Φ_{12} is optimal for English. In addition, the model Φ^p_{11} gives reasonable performance for both languages (and even outperforms Φ^p_{12} with respect to labeled exact match for English). We will therefore keep all three models when we go on to add dependency features in the next section.

5.3.2 Dependency Structure

The use of dynamic dependency type features for making parsing decisions is largely unchartered territory in the literature, which is due to the fact that data-driven dependency parsers normally do not construct labeled dependency graphs and therefore do not have access to dependency type labels during parsing. In this section, we investigate the effect of adding to the part-of-speech models the previously assigned dependency types of the top token ($d(\sigma_0)$), its leftmost and rightmost dependents ($d(l(\sigma_0))$, $d(r(\sigma_0))$), and the leftmost dependent of the next token ($d(l(\tau_0))$) (cf. section 4.2.4). The results are shown in table 5.3.

The most important observation is that adding the dependency type of the top token (Φ^d_{000}) gives a substantial increase in parsing accuracy across the board, for both languages, all part-of-speech models, and all evaluation metrics. Adding information about other dependencies has a more marginal impact, and the leftmost dependent of the top token even has a negative effect on attachment score for English. By and large, however, the models that incorporate all dependency features (Φ^d_{111}) are the best performing models for both languages, regardless of part-of-speech context, a result that holds without exception for the exact match criterion.

These results can be related to research on constituency-based parsing in the following way. The positive effect of including the dependency type of

Table 5.3. Accuracy as a function of dependency type features; AS: attachment score, EM: exact match; U: unlabeled, L: labeled; Swedish (cross-validation), English (section 00)

Model	Swedish				English			
	AS		EM		AS		EM	
	U	L	U	L	U	L	U	L
Φ^p_{11}	77.9	70.9	27.2	18.0	77.7	74.8	13.4	10.2
$\Phi^p_{11} + \Phi^d_{000}$	80.3	72.5	29.2	19.0	80.9	77.7	16.3	11.7
$\Phi^p_{11} + \Phi^d_{100}$	80.3	72.8	29.6	20.3	80.5	77.5	16.3	11.8
$\Phi^p_{11} + \Phi^d_{010}$	81.2	73.4	31.6	19.9	81.2	78.1	18.1	12.9
$\Phi^p_{11} + \Phi^d_{001}$	81.6	73.9	31.1	19.8	**81.8**	**78.8**	18.9	13.9
$\Phi^p_{11} + \Phi^d_{111}$	**82.3**	**74.9**	**33.6**	**22.6**	81.6	78.7	**19.5**	**15.0**
Φ^p_{21}	78.4	71.0	28.4	18.6	77.7	74.8	14.2	10.7
$\Phi^p_{21} + \Phi^d_{000}$	80.3	72.4	30.0	19.3	80.6	77.3	16.6	12.0
$\Phi^p_{21} + \Phi^d_{100}$	80.2	72.6	29.5	20.0	80.3	77.2	16.4	12.1
$\Phi^p_{21} + \Phi^d_{010}$	81.4	73.5	32.0	20.2	81.2	77.9	19.0	13.4
$\Phi^p_{21} + \Phi^d_{001}$	81.6	73.8	31.6	20.0	81.3	78.3	19.2	14.4
$\Phi^p_{21} + \Phi^d_{111}$	**82.2**	**74.8**	**33.4**	**22.1**	**81.5**	**78.5**	**19.8**	**15.2**
Φ^p_{12}	77.4	70.1	26.6	17.8	79.0	76.1	14.4	10.0
$\Phi^p_{12} + \Phi^d_{000}$	80.4	72.4	29.2	18.9	82.4	79.2	18.4	12.4
$\Phi^p_{12} + \Phi^d_{100}$	80.5	72.8	29.1	19.7	82.3	79.3	19.2	14.1
$\Phi^p_{12} + \Phi^d_{010}$	81.5	73.6	31.9	19.9	83.1	80.0	20.5	14.7
$\Phi^p_{12} + \Phi^d_{001}$	81.8	73.9	31.4	20.0	**83.5**	**80.5**	21.2	15.0
$\Phi^p_{12} + \Phi^d_{111}$	**82.5**	**75.1**	**33.5**	**22.2**	83.4	**80.5**	**21.9**	**17.0**

the top token mirrors the observation that grandparent nodes are important in phrase structure parsing (Collins, 1999; Charniak, 2000). Both types contribute to disambiguation by clarifying the grammatical function of the node in question. For example, observing that the top token has the dependency type *subject* or *object* can be helpful in exactly the same way as knowing that a particular NP is immediately dominated by an S node or a VP node in phrase structure parsing. The mixed evidence concerning dependents of the target nodes, in particular the rightmost dependent of the top token and the leftmost dependent of the next token, has bearing on the issue of whether it is relevant to consider sibling nodes in phrase structure parsing. For instance, while Charniak (2000) conditions on up to four preceding siblings, Collins (1997, 1999) achieves similar accuracy without including any information of this kind.

Comparing the different part-of-speech models, we see that the model Φ^p_{12} still gives the highest accuracy for English. However, after the addition of

Table 5.4. Accuracy as a function of part-of-speech context (with dependency type features); AS: attachment score, EM: exact match; U: unlabeled, L: labeled; Swedish (cross-validation), English (section 00)

Model	Swedish				English			
	AS		EM		AS		EM	
	U	L	U	L	U	L	U	L
$\Phi^p_{12} + \Phi^d_{111}$	**82.5**	**75.1**	**33.5**	**22.2**	83.4	80.5	21.9	17.0
$\Phi^p_{13} + \Phi^d_{111}$	82.4	74.9	33.2	21.9	**83.8**	**80.8**	22.6	17.1
$\Phi^p_{14} + \Phi^d_{111}$	82.1	74.5	32.5	21.3	83.6	80.7	**23.0**	**17.3**
$\Phi^p_{15} + \Phi^d_{111}$	82.0	74.5	31.6	20.9	83.5	80.6	22.8	17.2
$\Phi^p_{16} + \Phi^d_{111}$	82.0	74.4	31.4	20.5	83.7	**80.8**	22.7	17.0

the dependency type features, this model also gives the highest accuracy for Swedish with respect to attachment score. And with respect to exact match, it is now the model Φ^p_{11}, not Φ^p_{21}, that gets the top score. This result can probably be explained by the fact that the head of the top token, if present in a given configuration, is always identical to the second topmost token on the stack, which means that there is considerable redundancy in the information given by the features $p(\sigma_1)$ and $d(\sigma_0)$. In order to further investigate the interaction of part-of-speech features and dependency type features, we have also repeated part of the experiment presented in table 5.2, increasing the lookahead with respect to part-of-speech features, but this time in the presence of all dependency type features. The results are given in table 5.4.

For English, it is no longer detrimental to increase the lookahead. The best performing models now include three (Φ^p_{13}) or four (Φ^p_{14}) extra input tokens, and it is possible to increase the lookahead up to six tokens without more than marginal degradation. This seems to indicate that the positive effect of an increased lookahead is dependent on having a more richly articulated feature model to start from. For Swedish, we can observe similar results, although the best performing model is still limited to two lookahead tokens, and the degradation with increasing lookahead is somewhat steeper. This can probably be explained as a problem of sparse data, since the Swedish training corpus is one order of magnitude smaller than the English one. On the whole, however, we see very small differences as a function of varying the lookahead above one token.

5.3.3 Lexicalization

Lexicalization is usually considered a necessary condition for accurate disambiguation in data-driven parsing. Table 5.5 shows the result of adding lexical features to a subset of the models considered in previous sections. The upper

Table 5.5. Accuracy as a function of lexical features (with part-of-speech context and dependency features); AS: attachment score, EM: exact match; U: unlabeled, L: labeled; Swedish (cross-validation), English (section 00)

Model	Swedish				English			
	AS		EM		AS		EM	
	U	L	U	L	U	L	U	L
$\Phi^p_{12} + \Phi^d_{111}$	82.5	75.1	33.5	22.2	83.4	80.5	21.9	17.0
$\Phi^p_{12} + \Phi^d_{111} + \Phi^w_{10}$	83.7	78.5	34.9	26.3	85.2	83.2	25.7	22.5
$\Phi^p_{12} + \Phi^d_{111} + \Phi^w_{01}$	84.6	78.7	37.2	25.3	85.0	82.5	25.7	20.2
$\Phi^p_{12} + \Phi^d_{111} + \Phi^w_{11}$	85.6	81.5	39.1	30.2	86.6	84.8	29.9	26.2
$\Phi^p_{12} + \Phi^d_{111} + \Phi^w_{21}$	85.7	81.5	39.4	30.3	86.9	85.1	30.1	26.4
$\Phi^p_{12} + \Phi^d_{111} + \Phi^w_{12}$	**85.9**	**81.7**	39.4	30.1	**87.3**	85.4	30.1	26.2
$\Phi^p_{12} + \Phi^d_{111} + \Phi^w_{22}$	**85.9**	81.6	**39.8**	**30.4**	**87.3**	**85.6**	**31.1**	**27.7**
$\Phi^p_{11} + \Phi^d_{111} + \Phi^w_{22}$	**85.9**	**81.7**	39.5	**30.4**	86.3	84.6	28.8	25.7
$\Phi^p_{12} + \Phi^d_{111} + \Phi^w_{22}$	**85.9**	81.6	**39.8**	**30.4**	87.3	85.6	31.1	**27.7**
$\Phi^p_{13} + \Phi^d_{111} + \Phi^w_{22}$	85.8	81.5	39.6	30.0	**87.6**	**85.8**	**31.2**	27.3
$\Phi^p_{14} + \Phi^d_{111} + \Phi^w_{22}$	85.7	81.4	39.1	29.6	87.5	85.7	31.0	27.0
$\Phi^p_{15} + \Phi^d_{111} + \Phi^w_{22}$	85.5	81.1	39.0	29.4	87.5	85.7	31.0	26.8
$\Phi^p_{16} + \Phi^d_{111} + \Phi^w_{22}$	85.4	80.9	38.6	29.0	87.4	85.7	30.2	26.3

part of the table is based on the model $\Phi^p_{12} + \Phi^d_{111}$, incorporating all dependency type features and a lookahead of two tokens, and shows the effect of adding the word form of the top token (feature $w(\sigma_0)$, model Φ^w_{10}), the word form of the next token (feature $w(\tau_0)$, model Φ^w_{01}), and the word forms of both target tokens (model Φ^w_{11}), followed by the further addition of the head of the top token (feature $w(h(\sigma_0))$, model Φ^w_{21}), one lookahead token (feature $w(\tau_1)$, model Φ^w_{12}), and both of these tokens (model Φ^w_{22}) (cf. section 4.2.4).

First of all, we see that the benefit of lexicalization is evident for inductive dependency parsing as well as for other data-driven approaches. With very few exceptions, parsing accuracy increases steadily with the addition of each lexical feature, although the magnitude of the improvement quickly decreases. However, there is a clear difference in this respect between Swedish, where improvement is marginal for any features after the first two, and English, where accuracy increases more steadily, especially with respect to the exact match metrics. Again, this difference can probably be explained by reference to data sparseness for Swedish, which is even more sensitive for lexical features than for part-of-speech features.

The lower part of table 5.5 again explores the variation of lookahead, while keeping the number of lexical features constant at the maximum (Φ^w_{22}). We see that increasing the lookahead to three tokens is beneficial for English but

not for Swedish, while decreasing it to one has a strong negative effect for English but is barely noticeable for Swedish, where the labeled attachment score even goes up. These results give further support to the assumption that the Swedish data is too sparse to make effective use of the discriminative power of a more complex feature model.

Let us summarize the results concerning the influence of the feature model on parsing accuracy. For part-of-speech features, it is essential to include a context of at least one token in each direction, in addition to the top token and the next token. Increasing the context further on the stack side appears to give no improvement, while increasing the lookahead is beneficial to the extent that there are sufficient quantities of training data available. Dependency type features have a positive influence on parsing accuracy, especially with respect to the exact match metrics, and the dependency type of the top token is the single most important feature. Lexicalization, finally, has a substantial positive effect on parsing accuracy, but the addition of lexical features over and above the word form of the target tokens is dependent on the availability of large quantities of training data.

Given the results so far, we will now restrict our attention to a subset of the models considered, which will be evaluated with respect to efficiency and learning curves. In section 5.4, an even smaller subset will be used as a basis for the exploration of learning algorithm parameters. Altogether, we will consider five models, in order of increasing complexity:

$$
\begin{aligned}
B &= \Phi^{p}_{00} \\
P &= \Phi^{p}_{12} \\
D &= \Phi^{p}_{12} + \Phi^{d}_{111} \\
L_2 &= \Phi^{p}_{12} + \Phi^{d}_{111} + \Phi^{w}_{11} \\
L_4 &= \Phi^{p}_{12} + \Phi^{d}_{111} + \Phi^{w}_{22}
\end{aligned}
$$

The first model (B) is the baseline model, where the only features included are the parts-of-speech of the target tokens. The second model (P) is the best pure part-of-speech model for English, with a context of one stack token and two lookahead tokens. The third model (D) is the model obtained by adding all dependency type features to the P model. The last two models (L_2 and L_4) are lexicalized models, based on the D model, and incorporating two and four lexical features, respectively. Note that L_4 is not the best performing model for English, where the model $\Phi^{p}_{13} + \Phi^{d}_{111} + \Phi^{w}_{22}$ has a higher accuracy on the validation set. We will return to this model in the final evaluation in section 5.5. Table 5.6 gives an overview of the accuracy of the five selected models.

5.3.4 Efficiency

One of the the main tenets of this study is that, even though accuracy is the single most important evaluation criterion in natural language parsing,

Table 5.6. Accuracy as a function of feature model (selection); AS: attachment score, EM: exact match; U: unlabeled, L: labeled; Swedish (cross-validation), English (section 00)

Model	Swedish				English			
	AS		EM		AS		EM	
	U	L	U	L	U	L	U	L
B	71.0	64.5	15.6	12.3	58.4	56.2	3.7	3.1
P	77.4	70.1	26.6	17.8	79.0	76.1	14.4	10.0
D	82.5	75.1	33.5	22.2	83.4	80.5	21.9	17.0
L_2	85.6	81.5	39.1	30.2	86.6	84.8	29.9	26.2
L_4	85.9	81.6	39.8	30.4	87.3	85.6	31.1	27.7

it cannot be regarded in isolation from other requirements, such as robustness, disambiguation and efficiency. We have chosen to treat robustness and disambiguation as absolute requirements, but we still have to consider the trade-off between accuracy and efficiency. Increasing model complexity tends to correlate positively with parsing accuracy (if overfitting can be avoided) but negatively with efficiency. It is therefore important to see how the differences in accuracy observed so far correlate with differences in efficiency.

Table 5.7 shows the evaluation of our five selected models with respect to the efficiency metrics defined in section 5.1.1. The first column reports training time (T); the next three columns show parsing time, expressed as total parsing time (P), mean parsing time per sentence (S), and mean number of words parsed per second (W); the fifth column gives the memory consumption (M). Time is expressed in seconds for T and P, milliseconds for S, and memory consumption is reported in megabytes. It should be remembered that the size of both training and test corpora are about one order of magnitude larger for English than for Swedish (cf. table 5.1). (The results for Swedish are as usual the mean score of a nine-fold cross-validation.)

Starting with the training time, we see that the memory-based approach is very efficient, as can be expected, with training times of three to four minutes for the most complex models on section 2-21 of the Wall Street Journal section of the Penn Treebank. We also see that training time scales very well, with an approximately linear growth both with respect to the number of features (2, 5, 9, 11, and 13 for the selected models) and with respect to the size of the training corpus (one order of magnitude difference between Swedish and English).

With respect to parsing efficiency, the memory-based approach has a drawback in that all training instances have to be kept in memory during parsing. Even with the optimized storing and indexing provided by TiMBL, this is bound to show up in the measurements of time and memory consumption. Nevertheless, we see that although the English parsers generally require more

Table 5.7. Efficiency as a function of feature model; T: training time (s), P: parsing time (s), S: mean parsing time per sentence (ms), W: mean number of words parsed per second, M: memory requirements during parsing (MB)

Model	Swedish					English				
	T	**P**	**S**	**W**	**M**	**T**	**P**	**S**	**W**	**M**
B	4.1	3.4	5.4	2753.7	4	42.5	16.3	8.5	2844.5	4
P	7.0	5.3	8.4	1771.7	14	73.6	63.4	33.0	732.8	61
D	9.9	11.3	17.9	831.7	19	109.7	72.2	37.6	643.1	96
L_2	14.8	158.7	251.5	59.2	42	178.2	971.8	505.9	47.8	298
L_4	18.2	549.3	870.5	17.1	56	226.8	2861.3	1489.5	16.2	420

memory than the Swedish ones, the ratio is always less than 10:1 for the same feature model. And with respect to parsing time, there is very little difference at all when considering the mean number of words per second, except for the model P, which is surprisingly slow for English. The difference in parsing time per sentence can largely be explained by the fact that the mean sentence length is longer for English, and the difference in total parsing time of course by the different sizes of the test corpora.

The picture that clearly emerges from table 5.7 is that model complexity is the most important factor with respect to parsing time. For both languages, we see that parsing speed drops by one order of magnitude with the introduction of lexical features. Comparing total parsing times, the ratio between L_2 and D is about 14:1 for both languages. The crucial difference between lexicalized and non-lexicalized models is not the number of features, but the number of values per feature. Whereas both part-of-speech features and dependency features have value sets with a cardinality ranging from about 10 to 50, the number of values for lexical features are in the order of 10^4–10^5. Adding more lexical features slows down parsing even further, as can be seen in the difference between L_4 and L_2, which is about 3:1 and which is caused by the addition of two more lexical features. This can be compared with the difference between D and P, which is about 2:1 for Swedish and barely noticeable for English, despite the addition of four new features.

The decrease in efficiency with increasing model complexity has to be seen in relation to the gain in accuracy. Going back to table 5.6, we note that the addition of four dependency features from the P model to the D model gives a gain in accuracy of 4–5 percentage points for attachment score, and even more for exact match, with less than a 50% drop in parsing speed. By contrast, the addition of two more lexical features from the L_2 model to the L_4 model improves attachment score by 0.1–0.8 percentage points while reducing parsing speed by about 70%. Whether we are willing to pay this price or not is dependent on the requirements of our applications, which may put different constraints on the lowest acceptable accuracy or efficiency, but it illustrates

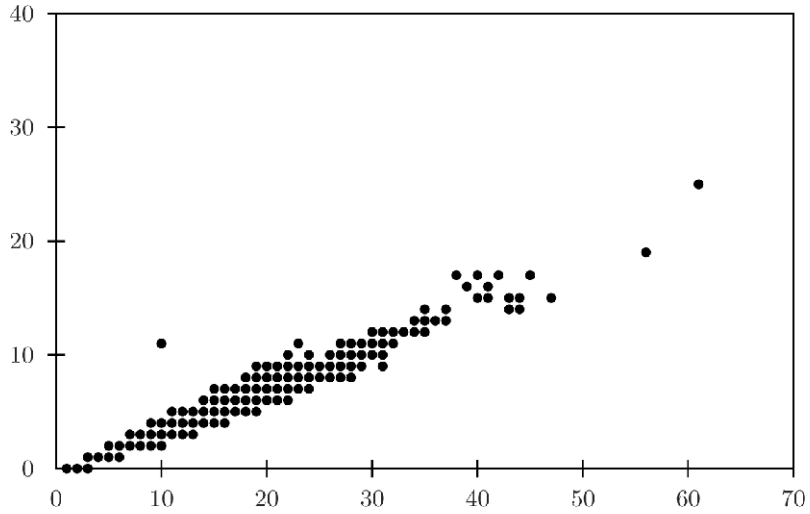

Fig. 5.10. Parsing time (ms) as a function of sentence length (number of words); Model B; Swedish, section 9 (630 data points)

the inevitable trade-off between accuracy and efficiency in data-driven parsing and the consequent need for joint optimization.

To complete the analysis of efficiency, we will consider the relation between sentence length and parsing time. In section 3.4, we established that the time complexity of the parsing algorithm is $O(n)$, where n is the number of words in the sentence, provided that transitions can be performed in constant time. Using a memory-based classifier to predict the next transition allows us to perform transitions in time that is constant in the number of words for a given feature model and training data set, but the size of this constant clearly depends on the complexity of the feature model and the size of the training data set. Moreover, because of the optimized storage and indexing techniques used in TiMBL, classifying a new instance may or may not require an exhaustive search of the instance base, which means that we can expect a larger variation in classification time for more complex models.

Figures 5.10–5.12 show a plot of parsing time (ms) as a function of sentence length for the models B, D and L_2 applied to section 9 of the Swedish treebank (after training on sections 1–8). The plot is based on 630 data points, each point representing one sentence. Figures 5.13–5.15 show the same plots for section 00 of the English treebank (after training on section 02–21 as usual), this time including 1920 sentences.

The linear behavior of the parser is most clearly discernible for the B model, where the time spent on classification is relatively small but also relatively constant given the very simple feature model. With increasing model complexity, the correlation between sentence length and parsing time becomes

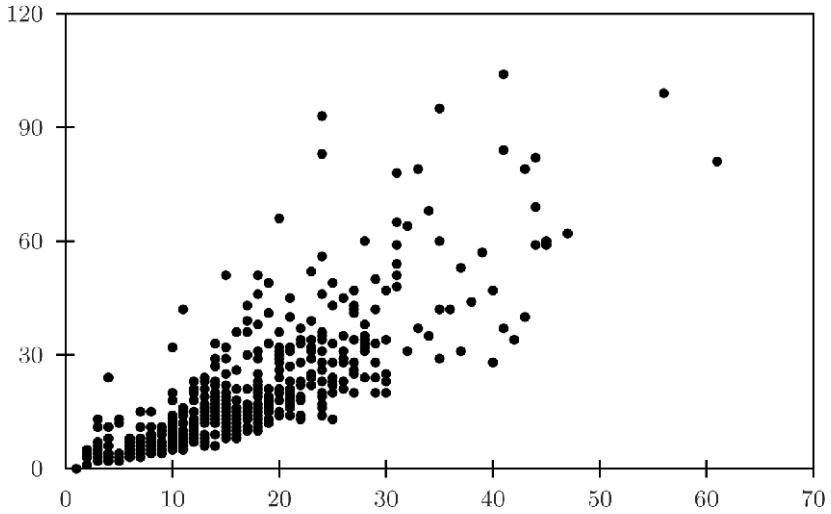

Fig. 5.11. Parsing time (ms) as a function of sentence length (number of words); Model D; Swedish, section 9 (630 data points)

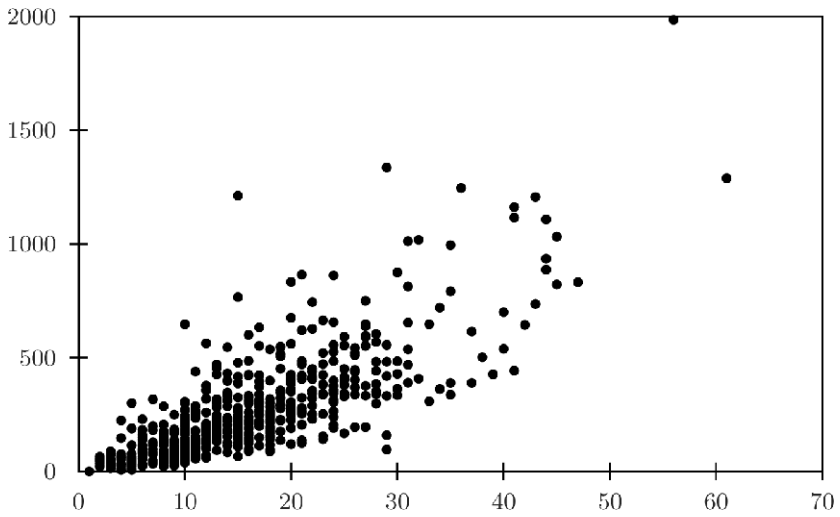

Fig. 5.12. Parsing time (ms) as a function of sentence length (number of words); Model L_2; Swedish, section 9 (630 data points)

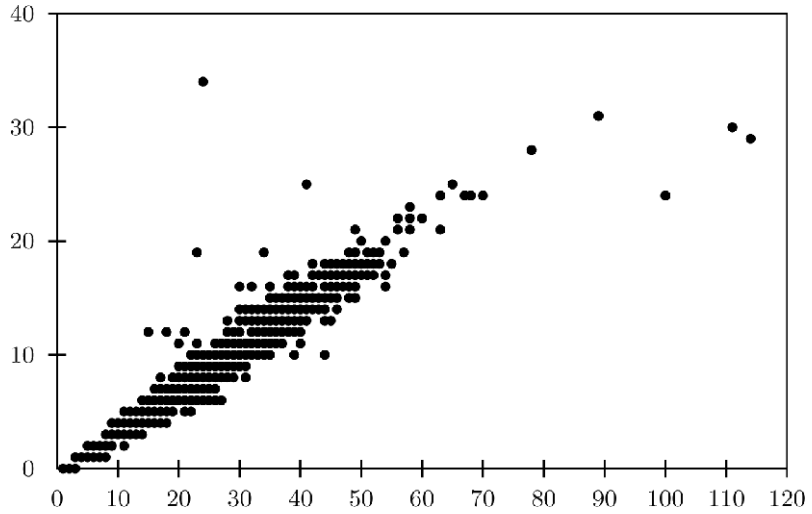

Fig. 5.13. Parsing time (ms) as a function of sentence length (number of words); Model *B*; English, section 00 (1920 data points)

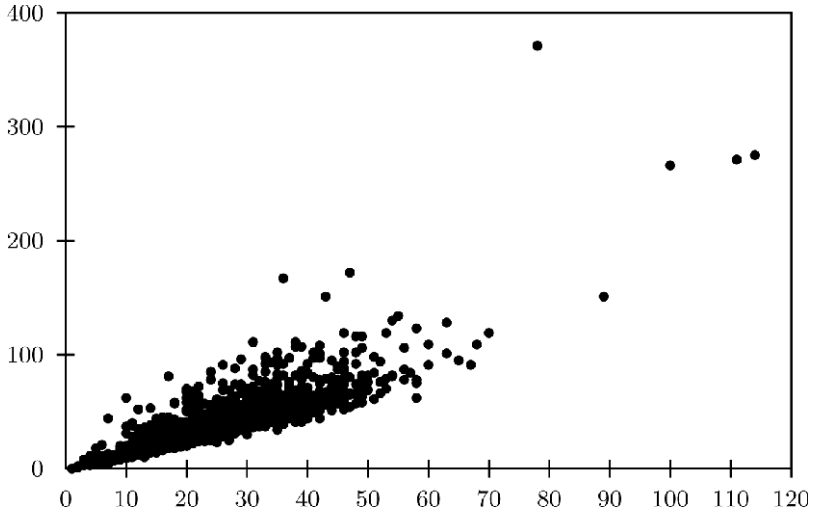

Fig. 5.14. Parsing time (ms) as a function of sentence length (number of words); Model *D*; English, section 00 (1920 data points)

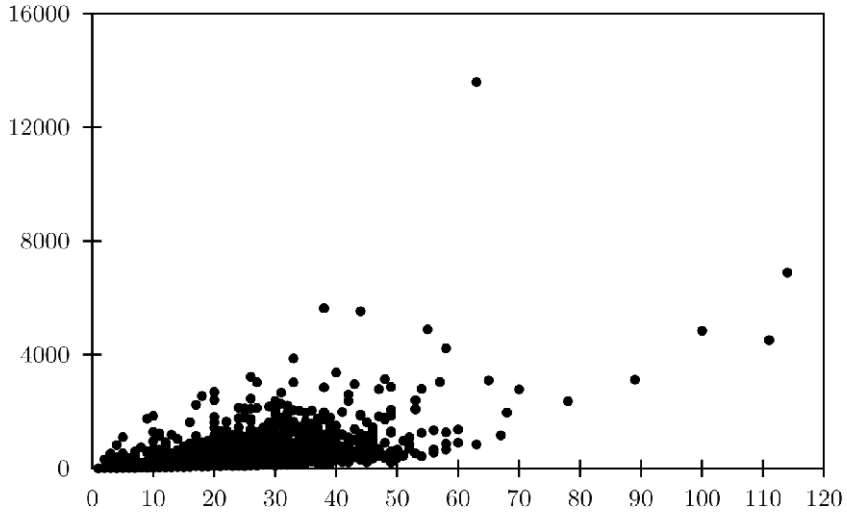

Fig. 5.15. Parsing time (ms) as a function of sentence length (number of words); Model L_2; English, section 00 (1920 data points)

increasingly noisy, as classification time begins to dominate parsing time. Even though the plots for some of the more complex models do not quite resemble straight lines, the data is very sparse for high values of the sentence length variable, and there are no grounds to reject the assumption that parsing time remains linearly related to sentence length even for more complex models, although the variance clearly increases due to the increased variance in classification time.

5.3.5 Learning Curves

The final aspect of feature models that will be considered in this evaluation is their sensitivity to the amount of training data available, which can be assessed by considering their learning curves. Figures 5.16–5.17 plot the accuracy of the feature models D and L_2 for Swedish as a function of the size of the training corpus. Figure 5.16 depicts the development of attachment score (labeled and unlabeled), while figure 5.17 shows exact match (labeled and unlabeled). The training corpus varies from 1 to 8 sections, and the values depicted are the mean of a nine-fold cross-validation as usual. Figures 5.18–5.19 give the same type of information for English, except that each increment of the training corpus represents two sections, i.e., one tenth of the entire training corpus, and measurements are only based on the validation set, section 00.

First of all, we may note that the lexicalized L_2 model generally has a steeper learning curve than the non-lexicalized D model, which is only to be expected given that the data is much more sparse for lexical features than

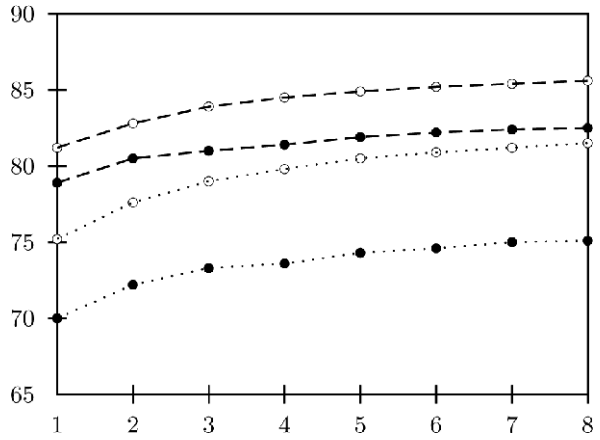

Fig. 5.16. Learning curve for attachment score; AS_U (dashed) and AS_L (dotted); models D (•) and L_2 (○); Swedish

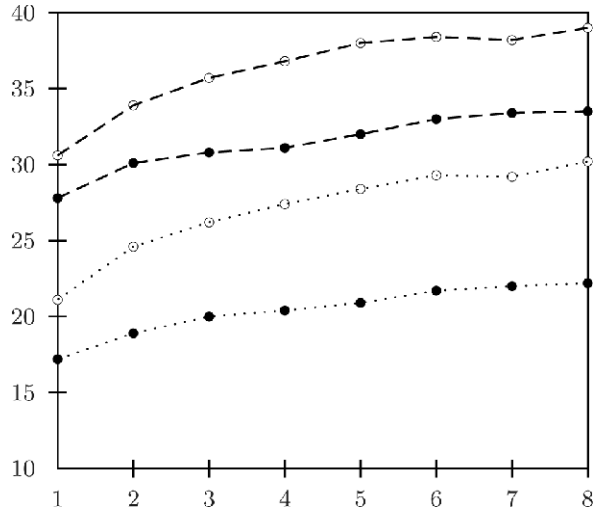

Fig. 5.17. Learning curve for exact match; EM_U (dashed) and EM_L (dotted); models D (•) and L_2 (○); Swedish

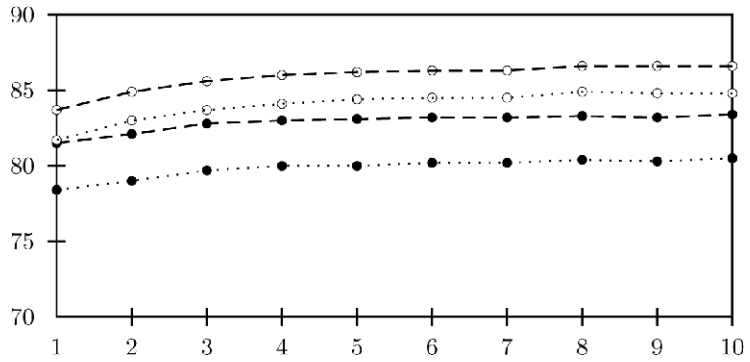

Fig. 5.18. Learning curve for attachment score; AS_U (dashed) and AS_L (dotted); models D (•) and L_2 (○); English

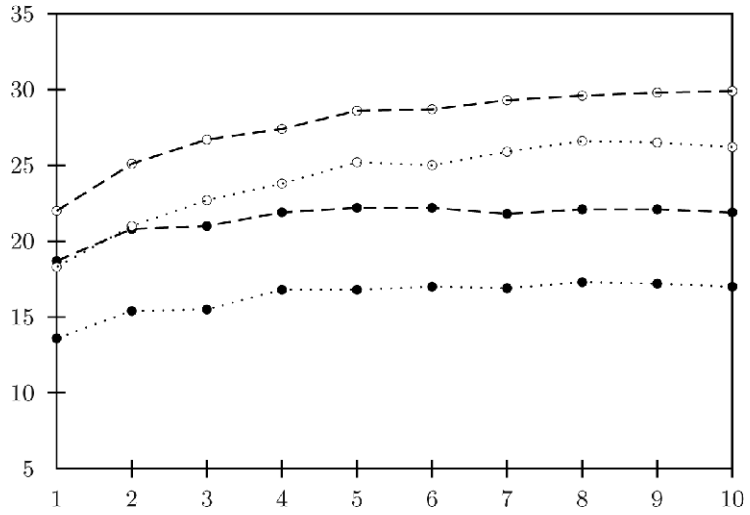

Fig. 5.19. Learning curve for exact match; EM_U (dashed) and EM_L (dotted); models D (•) and L_2 (○); English

for part-of-speech and dependency features. At the same time, it is worth pointing out that the L_2 model outperforms the D model at all data points, i.e., even when only training on a single section of the Swedish treebank, which only contains about 600 sentences and 10,000 tokens. This indicates that the analogy-based smoothing provided by the memory-based learner works well even for sparse data sets.

If we compare the two types of evaluation metrics, we see that exact match has a steeper learning curve than attachment score, which might be taken to show that the former is a more discriminative metric. Especially the L_2 model shows a steady increase in exact match accuracy for both languages.

Comparing the two languages, finally, it is evident that the curves are steeper for Swedish than for English. This is a direct reflection of the different sizes of the data sets involved, where the complete training corpus for Swedish is of about the same size as the smallest fraction considered for the English. In this way, the Swedish curves could almost be considered as a zoom-in of the first data point of the English curves. Some of the curves for English are really very flat, especially for attachment score and the D model. This is both good news and bad news. It is good news in the sense that (almost) optimal accuracy can often be obtained with only half of the training data, which can make parsing more efficient. But it is bad news in the sense that accuracy is unlikely to improve with the addition of more training data. The picture is somewhat different for the exact match metrics, where especially the L_2 model continues to improve up until the maximum size of the training corpus.

5.4 Learning Algorithm Parameters

Having examined the influence of different feature models on both accuracy and efficiency, we will now turn to the role of the learning algorithm, which in our case means exploring the parameter space of memory-based learning and classification, as provided by the TiMBL system. It is important to remember that feature selection and parameter optimization can be highly dependent on each other (Daelemans and Hoste, 2002; Daelemans et al., 2003). In principle, we should therefore explore the combined space of feature models and learning algorithm parameters in order to arrive at a truly optimal combination. In practice, this is often impossible because of the combinatorial effect, and the standard approach is therefore to keep one factor constant while varying the other systematically, which is also the methodology adopted in this study. However, because the factors are not independent, we need to make sure that the factor kept constant has a nearly optimal value, or the results can be highly misleading. This is the reason that we did not use the default settings of TiMBL when exploring different feature models in the preceding section, but instead used parameter settings that had been found optimal in previous studies. By the same reasoning, we will not use arbitrary feature models as our basis for parameter optimization in this section, but will limit our attention

to two of the best models. More precisely, we will study the D model, which was the best performing non-lexicalized model in the preceding section, and the L_2 model, which was a model that gave close to optimal performance for both Swedish and English and that strikes a good balance between accuracy and efficiency.

5.4.1 Neighbor Space and Distance Metric

Two fundamental parameters of the k-NN classification provided by memory-based learning are the number k of nearest neighbors (distances) taken into account and the metric used to compute the distance between instances. In our first experiment, we compare the Overlap and MVDM metrics, while varying the value of k from 1 to 9 in increments of 2.[2] The results are shown in table 5.8.

Starting with the Overlap metric, we see that performance is reasonable but not optimal with a k value of 1 but degrades drastically as the k value increases. The reason for this behavior is that, with the simple Overlap metric, there will typically be an abundance of ties in nearest neighbor position, and increasing the k value will lead to larger and larger portions of the instance base being used to classify each instance (Daelemans and Van den Bosch, 2005). This means that the set of local neighborhood approximations will eventually give way to a globally defined approximation, which simply assigns the overall majority class to every instance.

In terms of dependency parsing, this means that the parser will select the SHIFT transition in every configuration, constructing a dependency graph where every token is attached to the special root node. The attachment score of such a parser is simply the proportion of tokens attached to the special root node in the gold standard treebank, which happens to be 7.0% for the Swedish treebank and 4.7% for the English treebank. The non-lexicalized D model, which has fewer features and fewer values per feature, reaches this stage already at $k = 7$, while the lexicalized L_2 model degrades at a slightly slower pace. The degradation is accelerated by the fact that TIMBL uses the k nearest distances, rather than the k nearest neighbors, but the result will eventually be the same anyway.

Using the more sophisticated MVDM metric gives a completely different picture. Increasing the k value from 1 to 3 has a positive effect for both models with respect to all metrics, but further changes have very little effect overall. The highest accuracy is found at $k = 3$ for the D model in both languages. For the L_2 model, the optimum appears to be $k = 5$ for Swedish and $k = 7$ for English, although the differences are generally small. In the following experiments, we will therefore use $k = 3$ for the D model but use both $k = 5$ and $k = 7$ for the L_2 model.

[2] In order to avoid ties, it is recommended to use odd integers as values of k.

Table 5.8. Accuracy as a function of distance metric (Overlap, MVDM) and k value; AS: attachment score, EM: exact match; U: unlabeled, L: labeled; Swedish (cross-validation), English (section 00)

Model	**Metric**	k	**Swedish**				**English**			
			AS		**EM**		**AS**		**EM**	
			U	**L**	**U**	**L**	**U**	**L**	**U**	**L**
D	Overlap	1	**78.4**	**70.0**	**29.6**	**20.0**	**81.9**	**78.8**	**20.4**	**15.4**
		3	63.0	52.4	18.4	12.0	65.4	61.1	7.1	4.5
		5	31.7	24.1	7.4	4.9	37.7	34.3	1.2	1.0
		7	7.0	7.0	3.8	3.8	4.7	4.7	0.4	0.4
		9	7.0	7.0	3.8	3.8	4.7	4.7	0.4	0.4
	MVDM	1	80.8	72.7	31.8	20.9	82.9	79.9	20.9	16.1
		3	**82.5**	75.0	**33.1**	**21.5**	**83.4**	**80.5**	**21.9**	**17.1**
		5	**82.5**	**75.1**	32.8	20.7	83.1	80.1	21.8	16.5
		7	**82.5**	**75.1**	33.0	21.0	82.8	79.8	21.7	16.1
		9	82.2	74.8	32.3	20.7	82.7	79.6	20.9	15.0
L_2	Overlap	1	**82.2**	**76.2**	**34.3**	**24.7**	**85.2**	**83.0**	**27.0**	**22.5**
		3	72.5	63.5	22.8	14.8	74.1	71.1	12.0	9.0
		5	44.9	34.8	11.2	6.7	51.0	47.5	2.7	1.5
		7	9.5	8.9	4.0	3.8	19.7	18.4	0.5	0.4
		9	7.0	7.0	3.8	3.8	4.7	4.7	0.4	0.4
	MVDM	1	83.1	78.6	34.9	26.0	85.1	83.2	26.2	23.4
		3	84.7	80.6	37.2	28.6	86.2	84.5	29.2	25.8
		5	**84.8**	**80.8**	**37.6**	**29.1**	**86.3**	84.5	**29.7**	25.2
		7	84.7	80.7	37.4	29.0	**86.3**	**84.6**	**29.7**	**26.3**
		9	84.7	80.6	37.1	28.8	86.2	84.4	29.6	26.0

While the MVDM metric is in most cases superior to the Overlap metric already at $k = 1$ and improves with larger k values, it is also more sensitive to data sparseness. TiMBL therefore provides a back-off from MVDM to Overlap through a frequency threshold l, which means that the distance metric switches from MVDM to Overlap whenever one or both values compared occur less than l times in the training data. Table 5.9 shows the effect of increasing this threshold from 1 to 5 in increments of 1 for the D model with $k = 3$ and the L_2 model with $k = 5$ and $k = 7$. We see that the D model is completely insensitive to this parameter, with identical results for all settings, indicating that data sparseness is not a problem for the non-lexicalized D model, not even for Swedish. For the L_2 model, we see a small but steady improvement with a higher threshold, especially for Swedish with a more limited amount of training data. For $k = 5$ the optimal threshold appears to be 3 for both

Table 5.9. Accuracy as a function of switching threshold l (MVDM to Overlap); AS: attachment score, EM: exact match; U: unlabeled, L: labeled; Swedish (cross-validation), English (section 00)

Model	k	l	Swedish				English			
			AS		EM		AS		EM	
			U	L	U	L	U	L	U	L
D	3	1	82.5	75.0	33.7	21.5	83.4	80.5	21.9	17.1
		2	82.5	75.0	33.7	21.5	83.4	80.5	21.9	17.1
		3	82.5	75.0	33.7	21.5	83.4	80.5	21.9	17.1
		4	82.5	75.0	33.7	21.5	83.4	80.5	21.9	17.1
		5	82.5	75.0	33.7	21.5	83.4	80.5	21.9	17.1
L_2	5	1	84.8	80.8	37.6	29.1	86.3	84.5	29.7	25.9
		2	85.2	81.1	38.4	29.3	**86.5**	84.7	29.9	**26.1**
		3	85.3	**81.2**	**38.8**	**29.7**	**86.5**	**84.8**	29.9	25.9
		4	**85.4**	81.1	**38.8**	29.6	**86.5**	84.7	**30.0**	26.0
		5	85.3	81.0	38.5	29.3	**86.5**	84.7	**30.0**	25.8
	7	1	84.7	80.7	37.4	29.0	86.3	84.6	29.7	26.3
		2	85.0	80.9	38.2	29.1	86.5	84.7	29.9	**26.5**
		3	85.1	**81.0**	38.4	**29.2**	**86.6**	**84.8**	29.9	26.3
		4	**85.2**	**81.0**	**38.5**	**29.2**	**86.6**	84.6	29.9	26.2
		5	85.1	80.9	38.3	**29.2**	**86.6**	**84.8**	**30.0**	26.2

languages, which confirms earlier experiments (Nivre et al., 2004; Nivre and Scholz, 2004). For $k = 7$ the best results are obtained with a threshold of 5 for English, whereas for Swedish accuracy is consistently worse than for $k = 5$. In the following, we will therefore consider $k = 5$, $l = 3$ to be optimal for Swedish but consider both $k = 5$, $l = 3$ and $k = 7$, $l = 5$ for English.

5.4.2 Weighting Schemes

In addition to distance metric and k value, the k-NN classification may be tuned by different weighting schemes. In this section, we will consider two types of weighting. On the one hand, we have applied feature weighting with Information Gain (IG) and Gain Ratio (GR). On the other hand, we have used distance-weighted class voting with inverse distance (ID) and inverse-linear (IL) weighting. The results of these experiments are reported in table 5.10.

The overall tendency is that these weighting schemes have a negative influence on parsing accuracy. One possible explanation is that the MVDM metric in itself has a feature weighting effect, as observed in section 4.3, and that additional weighting will therefore result in overfitting. For the D model, the best results are obtained with IG feature weighting, which gives marginally

Table 5.10. Accuracy as a function of weighting scheme; FW: feature weighting (GR: gain ratio, IG: information gain); DWCV: distance weighted class voting (ID: inverse distance, IL: inverse linear); AS: attachment score, EM: exact match; U: unlabeled, L: labeled; Swedish (cross-validation), English (section 00)

Model	k	l	FW	DWCV	**Swedish**				**English**			
					AS		**EM**		**AS**		**EM**	
					U	**L**	**U**	**L**	**U**	**L**	**U**	**L**
D	3	1	–	–	**82.5**	75.0	**33.7**	21.5	**83.4**	**80.5**	21.9	**17.1**
			GR	–	82.1	74.9	32.9	**22.3**	83.0	80.1	21.7	17.0
			IG	–	82.2	**75.1**	33.0	21.5	**83.4**	**80.5**	**22.0**	**17.1**
			–	ID	82.0	74.4	33.3	21.8	**83.4**	80.4	21.7	**17.1**
			–	IL	81.0	73.0	32.2	21.3	82.9	79.9	21.5	16.6
L_2	5	3	–	–	85.3	81.2	38.8	29.7	86.5	**84.8**	**29.9**	25.9
			GR	–	84.6	80.4	37.1	28.7	86.0	84.1	29.0	25.5
			IG	–	83.8	79.8	34.6	26.7	86.0	84.5	27.3	24.4
			–	ID	**85.6**	**81.5**	**39.1**	**30.2**	**86.6**	**84.8**	**29.9**	**26.2**
			–	IL	85.0	80.6	38.1	28.9	86.0	84.2	28.3	25.0
	7	5	–	–	–	–	–	–	86.6	84.8	30.0	26.2
			GR	–	–	–	–	–	85.7	83.8	28.2	24.9
			IG	–	–	–	–	–	86.0	84.5	27.5	24.9
			–	ID	–	–	–	–	**86.8**	**85.0**	**30.3**	**26.9**
			–	IL	–	–	–	–	86.5	84.7	29.4	26.0

higher AS_L for Swedish and EM_U for English, but also lower AS_U and EM_U for Swedish. By an appeal to Occam's razor we will therefore conclude that the optimal parameter settings for non-lexicalized models are as follows:

1. Number of nearest distances: $k = 3$
2. Distance metric: MVDM with $l = 1$
3. Feature weighting: None
4. Distance-weighted class voting: None

Turning to L_2, we find that distance-weighted class voting with ID weighting has a consistent positive effect, whereas all the other weighting schemes are detrimental. We also see that, with the addition of ID weighting, $k = 7$ and $l = 5$ gives better performance than $k = 5$ and $l = 3$ for English. Hence, we conclude that the following settings are optimal for our lexicalized models:

1. Number of nearest distances: $k = 5$ (Swedish), $k = 7$ (English)
2. Distance metric: MVDM with $l = 3$ (Swedish), $l = 5$ (English)
3. Feature weighting: None
4. Distance-weighted class voting: ID

With the exception of the higher k and l values for English, these are also the settings that were used in the validation of feature models, and which have been found optimal in previous studies (Nivre et al., 2004; Nivre and Scholz, 2004).

5.5 Final Evaluation

In this section, we will assess the quality of the feature models and parameter settings that produced the best results during validation by applying them to an independent test set. In this way, we can hope to get a more unbiased estimate of the expected accuracy and efficiency with respect to new data. However, it is important to remember that, even though the test sets have not been used in the validation phase, they are nevertheless sampled from the same treebanks as the respective training and validation sets. Our estimates will therefore be valid for text that belongs to the same population, but not for text from other sources in Swedish or English.

We will also compare the performance of the best models to the state of the art in dependency-based parsing. This will raise certain questions about the sources of parsing errors, and we will try to tease apart the influence of three such sources: errors in part-of-speech tagging, errors in the function approximation, and errors due to the greedy, deterministic parsing strategy.

5.5.1 Accuracy and Efficiency

Table 5.11 shows the accuracy obtained on the test sets for the best lexicalized and non-lexicalized models, using the optimal parameter settings of the learning algorithm. We have included both the best models from section 5.3 and the models used as the basis for parameter optimization in section 5.4. This yields five models altogether, since the best non-lexicalized model for Swedish from section 5.3 is identical to the D model used in section 5.4. We use the notation D' and L'_4 to denote the models that are exactly like D and L_4 except that they include a part-of-speech lookahead of three tokens instead of two.

If we compare the results to those obtained during validation, they are comparable in all cases, which indicates that the models have not been overfitted to the training and validation data. Differences between validation scores and test scores are generally less than one percentage point for attachment scores, whereas the exact match scores show a variation of up to two percentage points (with a decrease in accuracy for Swedish and an increase for English). This is only natural, given that the number of observations is one order of magnitude greater for the attachment scores, which are word-based, than for the exact match scores, which are sentence-based.

As for the comparison between feature models, the only differences that are statistically significant for Swedish are those between the non-lexicalized

Table 5.11. Final evaluation: accuracy; AS: attachment score, EM: exact match; U: unlabeled, L: labeled; Swedish (section 0), English (section 23)

Model	Swedish				English			
	AS		EM		AS		EM	
	U	L	U	L	U	L	U	L
$D = \Phi^p_{12} + \Phi^d_{111}$	83.2	75.8	33.4	20.2	83.5	80.5	23.2	17.3
$D' = \Phi^p_{13} + \Phi^d_{111}$	83.3	75.7	31.8	19.2	83.6	80.7	23.5	18.0
$L_2 = \Phi^p_{12} + \Phi^d_{111} + \Phi^w_{11}$	**86.3**	**82.0**	37.7	29.6	87.4	85.7	30.8	26.8
$L_4 = \Phi^p_{12} + \Phi^d_{111} + \Phi^w_{22}$	86.1	81.8	37.3	29.8	87.8	86.0	32.7	**28.7**
$L'_4 = \Phi^p_{13} + \Phi^d_{111} + \Phi^w_{22}$	**86.3**	**82.0**	**39.2**	**30.8**	**88.1**	**86.3**	**32.8**	28.4

models D and D', on the one hand, and the lexicalized models L_2, L_4 and L'_4, on the other. Between these groups, the difference is statistically significant beyond the 0.01 level for all metrics (McNemar's test).[3] Within the groups, however, there are no significant differences.[4] For English we find basically the same pattern, but in addition to the differences between non-lexicalized and lexicalized models, which are all significant beyond the 0.0001 level, there are also significant differences between the model with two lexical features (L_2) and the models with four lexical features (L_4 and L'_4) with $p < 0.01$ for all metrics. Comparing L_4 and L'_4, finally, the differences appear to be significant for the attachment scores but not for the exact metrics. However, this result should be taken with a pinch of salt, since the attachment scores are based on observations of word tokens, which are very far from being independent of each other. Therefore, the lack of a significant difference in the sentence-based exact match comparison throws serious doubt on the value of the differences in attachment score. In conclusion, it therefore seems fair to say that the difference in accuracy between lexicalized and non-lexicalized models is statistically significant for both languages, and that the addition of two extra lexical features makes a significant difference for English, with the larger data sets, but not for Swedish. Any conclusions beyond these are not clearly warranted by the experimental results.

In order to get a more fine-grained picture of accuracy, we will now consider the accuracy for different dependency types. Table 5.12 gives unlabeled attachment score (AS_U), labeled precision (P), recall (R) and F measure (F) for the top scoring model L'_4 on the Swedish test set. Broadly speaking, we can divide dependency types according to accuracy into three sets. In the high-accuracy set, with a labeled F measure from 84% to 98%, we find all

[3] For EM_U $p = 0.01$; for the other three metrics $p < 0.0001$.

[4] It is worth remembering that the Swedish test set only contains 631 sentences, which means that the difference between 29.6% and 29.8% in the EM_L score for L_2 and L_4 is the difference of a single sentence.

Table 5.12. Final evaluation: attachment score (AS_U), precision (P), recall (R) and F measure per dependency type: Swedish (model L'_4)

Label	**n**	**AS_U**	**P**	**R**	**F**
ADV	1607	79.8	75.8	76.8	76.3
APP	42	23.8	38.1	19.0	25.4
ATT	950	81.3	79.9	78.5	79.2
CC	963	82.5	78.1	79.8	78.9
DET	947	92.6	88.9	90.2	89.5
ID	254	72.0	72.5	58.3	64.6
IM	133	98.5	98.5	98.5	98.5
INF	10	100.0	100.0	30.0	46.2
OBJ	585	88.0	78.2	77.3	77.7
PR	985	94.2	88.6	92.7	90.6
PRD	244	90.6	76.7	77.0	76.8
ROOT	607	91.3	84.6	91.3	87.8
SUB	957	89.8	86.7	82.5	84.5
UK	213	85.0	89.4	83.6	86.4
VC	238	93.7	82.1	90.6	86.1
XX	29	82.8	85.7	20.7	33.3
Total	8782	86.3	82.0	82.0	82.0

dependency types where the head is a closed class word: IM (marker → infinitive), PR (preposition → noun), UK (complementizer → verb) and VC (auxiliary verb → main verb). We also find the type DET (noun → determiner), which has similar characteristics although the determiner is not treated as the head in the Swedish annotation. The high-accuracy set also includes the central dependency types ROOT and SUB, which normally identify the finite verb of the main clause and the grammatical subject, respectively.

In the medium-accuracy set, with a labeled F measure in the range of 75–80%, we find the remaining major dependency types, ADV (adverbial), ATT (nominal modifier), CC (coordination), OBJ (object) and PRD (predicative). However, this set can be divided into two subsets, the first consisting of ADV, ATT and CC, which have an unlabeled attachment score not too much above the labeled F measure, indicating that parsing errors are mainly due to incorrect attachment. This is plausible also because ADV and ATT are the dependency types typically involved in modifier attachment ambiguities, and coordination is a source of attachment ambiguities as well. The second subset contains OBJ and PRD, which both have an unlabeled attachment score close to 90%, which means that they are often correctly attached but may be incorrectly labeled. This is again plausible, since these types identify nominal arguments of the verb (other than the subject), which can often occur in the same syntactic contexts.

Table 5.13. Final evaluation: attachment score (AS_U), precision (P), recall (R) and F measure per dependency type: English (model L'_4)

Label	**n**	**AS_U**	**P**	**R**	**F**
AMOD	2072	78.2	80.7	73.0	76.7
DEP	259	42.9	56.5	30.1	39.3
NMOD	21002	91.2	91.1	90.8	91.0
OBJ	1960	86.5	78.9	83.5	81.1
PMOD	5593	90.2	87.7	89.5	88.6
PRD	832	90.0	75.9	71.8	73.8
ROOT	2401	86.4	78.8	86.4	82.4
SBAR	1195	86.0	87.1	85.1	86.1
SBJ	4108	90.0	90.6	88.1	89.3
VC	1771	98.8	93.4	96.6	95.0
VMOD	8175	80.3	76.5	77.1	76.8
Total	49368	88.1	86.3	86.3	86.3

Finally, we have a low-accuracy set, with a labeled F measure below 70%, where the common denominator is mainly that these dependency types are rare: INF (infinitive complements), APP (appositions), XX (unclassifiable). The only exception to this generalization is the type ID (idiom constituent), which is not that rare but which is rather special for other reasons. All types in this set except APP have a relatively high unlabeled attachment score, but their labels are seldom used correctly. An extreme case is INF, which has both an unlabeled attachment score and a labeled precision of 100%, although there are only 10 instances in total in the test set, but which has a much lower labeled recall (30%). The dependency type APP, finally, is used for a family of loosely connected modifiers of either nouns or verbs, which apparently are very difficult to attach correctly.

Table 5.13 gives the same kind of breakdown across dependency types for the top scoring model L'_4 on the English test set, where we can distinguish a similar division into three sets according to accuracy level. In the high-accuracy set, with a labeled F measure from 86% to 95%, we find SBJ (subject) and three dependency types where the head is a closed class word: PMOD (preposition → complement/modifier), VC (auxiliary verb → main verb) and SBAR (complementizer → verb). In addition, this set includes the type NMOD, which includes the noun-determiner relation as an important subtype.

In the medium-accuracy set, with a labeled F measure from 74% to 82%, we find the types AMOD, VMOD, OBJ, PRD and ROOT. The former two dependency types mostly cover adverbial functions, and have a labeled accuracy not too far below their unlabeled attachment score, which is an indication that the main difficulty lies in finding the correct head. By contrast, the argument functions OBJ and PRD have a much better unlabeled attachment score,

which shows that they are often attached to the correct head but misclassified. This tendency is especially pronounced for the PRD type, where the difference is more than 15 percentage points, which can probably be explained by the fact that this type is relatively infrequent in the annotated English data.

The low-accuracy set for English only includes the default classification DEP. The very low accuracy for this dependency type can be explained by the fact that it is both a heterogeneous category and the least frequent dependency type in the data.

If we compare the results across languages, we can distinguish the following general patterns:

- Dependents of closed class words have high accuracy, labeled as well as unlabeled. This includes the following construction types:
  1. Preposition → Noun (Swedish PR, English PMOD[5])
  2. Complementizer → Verb (Swedish UK, English SBAR)
  3. Auxiliary verb → Main verb (Swedish and English VC)
- Core arguments of the verb have high unlabeled accuracy. This includes:
  1. Subjects (Swedish SUB, English SBJ)
  2. Objects (Swedish and English OBJ)
  3. Predicative complements (Swedish and English PRD)

  Subjects also have high labeled accuracy, whereas objects and predicative complements are more easily confused with each other.
- Modifiers generally have medium accuracy, both labeled and unlabeled.
- Atypical, heterogeneous and rare dependency types have low accuracy, especially when labels are taken into account.

One apparent difference between the languages is that nominal modifiers (NMOD) have a very high accuracy for English (90.5% AS_U, 90.4% F), whereas the closest corresponding dependency type for Swedish (ATT) has both unlabeled and labeled accuracy below 80%. However, this can probably be explained by the fact that the English category NMOD contains two prominent subcategories that contribute significantly to the overall result. These subcategories are determiners, which have a very high accuracy also in Swedish (91.5% AS_U, 89.6% F), and constituents of noun-noun compounds, which can usually be identified relatively easily but which are absent in the Swedish data because of different orthographic conventions.

Another difference is that the type ROOT, which normally identifies the finite verb of the main clause, has a considerably higher accuracy for Swedish, where it belongs to the high-accuracy set, than for English, where it is found in the middle set. This is probably related to the greater sentence complexity in the English data set, where a greater mean sentence length can be expected to correlate with a greater mean number of clauses per sentence, which tends to make the identification of the main clause more difficult.

[5] The PMOD type also contains modifiers of prepositions but is heavily dominated by nominal complements.

Table 5.14. Final evaluation: efficiency; T: training time (s), P: parsing time (s), S: mean parsing time per sentence (ms), W: mean number of words parsed per second, M: memory requirements during parsing (MB)

Model	Swedish					English				
	T	**P**	**S**	**W**	**M**	**T**	**P**	**S**	**W**	**M**
$D = \Phi^p_{12} + \Phi^d_{111}$	9.5	10.1	16.0	974.4	19	102.8	71.7	29.7	790.6	96
$D' = \Phi^p_{13} + \Phi^d_{111}$	12.0	13.6	21.6	723.6	26	118.9	93.7	38.8	605.0	155
$L_2 = \Phi^p_{12} + \Phi^d_{111} + \Phi^w_{11}$	16.0	167.0	264.7	58.9	45	171.0	1140.5	472.1	49.7	295
$L_4 = \Phi^p_{12} + \Phi^d_{111} + \Phi^w_{22}$	18.0	574.0	909.7	17.1	57	217.9	3327.8	1380.0	17.0	416
$L'_4 = \Phi^p_{13} + \Phi^d_{111} + \Phi^w_{22}$	20.9	661.2	1050.0	14.9	66	234.6	3641.8	1510.0	15.6	457

Finally, it is worth remembering that the two data sets differ in the treatment of coordination, which falls under a separate dependency type (CC) for Swedish, which is comparable in accuracy to the modifier constructions. For English, coordination is not analyzed as a separate category, which means that coordinate structures can be present in any category, although most instances are likely to be found in the VMOD and NMOD categories. Since the analysis assigned to coordinate structures in this way is often linguistically inadequate, it is fair to say that the accuracy results for English give a too optimistic estimate of the true accuracy for parsing unrestricted text.

To complete the picture on model assessment, we also present an evaluation of efficiency on the training and test sets. The results are presented in table 5.14. As can be expected, there are no significant deviations from the results obtained during validation. The higher absolute parsing times (P) for English are due to the fact that the test set is larger than the validation set, but the mean number of words parsed per second is very similar. Relating efficiency to accuracy, we may note that for Swedish the L_4 and L'_4 models more than triple the parsing time without a statistically significant improvement in accuracy, which makes the L_2 model appear as the best choice for a joint optimization of accuracy and efficiency. For English, the addition of two more lexical features gives a significant improvement in accuracy, which means that the more complex models will be optimal for applications where we can accept the decrease in parsing speed. Finally, it is worth pointing out that the differences observed between Swedish and English in this respect are more probably related to the size of the data sets than anything else.

5.5.2 Related Work

In this section, we will try to relate the results from the final evaluation to the state of the art in dependency-based text parsing. For Swedish this is rather difficult, since there is no comparable evaluation reported in the literature, let alone based on the same data. Most of the parsers developed for Swedish use

constituency-based representations, either for full parsing (Sågvall Hein, 1982) or partial parsing (Kokkinakis and Johansson Kokkinakis, 1999; Megyesi, 2002; Bigert, 2005). Voutilainen (2001) presents a partial and informal evaluation of a Swedish FDG parser, based on manually checked parses of about 400 sentences from newspaper text, and reports F measures of 95% for subjects and 92% for objects. These results clearly indicate a higher level of accuracy than that attained in the experiments reported here, but without knowing the details of the data selection and evaluation procedure it is very difficult to draw any precise conclusions. In any case, the results reported in this study may serve as a benchmark for future evaluations of Swedish dependency parsing. The results are encouraging, given the limited amount of data available for training, but it must also be kept in mind that the Swedish data set does not exhibit the same level of complexity as the English one.

For English there is much more relevant work to compare with. The first large-scale evaluation of dependency parsing on the Wall Street Journal data was performed by Eisner (1996b,a). However, Eisner excluded certain types of sentences from the evaluation, in particular sentences involving coordination, which means that the results are not strictly comparable. The same goes for the evaluations of grammar-driven parsers such as the statistical CDG parser of Wang and Harper (2004), the XLE LFG parser of Kaplan et al. (2004) and the CCG parser of Clark and Curran (2004), which are all based on different ways of extracting dependencies from the Penn Treebank data, the latter two using the PARC 700 Dependency Bank (King et al., 2003) and the CCGbank (Hockenmaier, 2003a), respectively.

By contrast, the results reported by Yamada and Matsumoto (2003) and Isozaki et al. (2004) are based on exactly the same data samples and conversion methods (except that they only consider unlabeled dependencies). In addition, Yamada and Matsumoto (2003) derive comparable results for the parsers of Collins (1997) and Charniak (2000), by applying the same conversion to the output of these parsers. More recently, the same data sets have also been used by McDonald, Crammer and Pereira (2005). Table 5.15 presents a comparison of our results with those obtained with the other systems, limited to unlabeled accuracy metrics. In addition to the usual metrics AS_U and EM_U, we also break down the attachment score into *dependency accuracy* (DA), which is the attachment score for all tokens not attached to the special root node, and *root accuracy* (RA), which is the attachment score for all tokens attached to the special root node. Given the conversion of the Penn Treebank annotation to dependency graphs, there is exactly one token per sentence attached to the special root node in the gold standard.

It is clear that, with respect to unlabeled accuracy, our parser does not quite reach state-of-the-art performance, even if we limit the competition to deterministic methods such as those of Yamada and Matsumoto (2003) and Isozaki et al. (2004). We believe that there may be three different reasons for this. First of all, the part-of-speech tagger used for preprocessing in our experiments has a lower accuracy than the one used by Yamada and Matsumoto

Table 5.15. Comparison with related work; AS_U: unlabeled attachment score, DA_U: dependency accuracy, RA_U: root accuracy, EM_U: unlabeled exact match

Study	**AS_U**	**DA_U**	**RA_U**	**EM_U**
Collins (1997) (Model 3)	91.7	91.5	95.2	43.3
Charniak (2000)	**92.2**	**92.1**	95.2	**45.2**
Yamada and Matsumoto (2003)	90.4	90.3	91.6	38.4
Isozaki et al. (2004)	91.4	91.2	**95.7**	40.7
McDonald, Crammer and Pereira (2005)	91.0	90.9	94.2	37.5
This study	88.1	88.2	86.4	32.8

(2003) (96.1% vs. 97.1%).[6] Although this is not a very interesting explanation, it undoubtedly accounts for part of the difference. We will return to this in our error analysis in the next section.

A more important factor is the relatively low root accuracy of our parser, which may reflect a weakness in the one-pass parsing strategy with respect to the global structure of complex sentences. Although the systems of Yamada and Matsumoto (2003) and Isozaki et al. (2004) are also deterministic, they perform multiple passes over the input, building the structures bottom-up. In the case of Isozaki et al. (2004), parsing is also preceded by a separate root detection phase, which explains the improved root accuracy of this system over Yamada and Matsumoto (2003).

It is noteworthy that our parser has lower root accuracy than dependency accuracy, whereas the inverse holds for all the other parsers. The problem becomes even more visible when we compare dependency and root accuracy for sentences of different lengths, as shown in table 5.16. Here we see that for really short sentences (up to 10 words) root accuracy is indeed higher than dependency accuracy, but while dependency accuracy degrades very gracefully with sentence length, the root accuracy drops more drastically (which also very clearly affects the exact match score). It is also interesting to compare with the results for Swedish, where the mean sentence length is considerably smaller, and where root accuracy is indeed as high as 91.3% (cf. table 5.12). This may be taken to suggest that some kind of preprocessing in the form of clausing or root detection could improve overall parsing accuracy.

Although it seems clear that the inferior root accuracy is primarily related to the single-pass deterministic parsing strategy, it is less clear whether the lower dependency accuracy is due to the parsing strategy or to lower prediction accuracy of the classifiers involved. Whereas our system uses memory-based learning and classification, the systems of Yamada and Matsumoto (2003) and Isozaki et al. (2004) are based on support vector machines. In a recent study by Sagae and Lavie (2005), using data from the Penn Treebank and

[6] Isozaki et al. (2004) apparently used the tags provided by the Collins parser, although this is not entirely clear from the description in the paper.

Table 5.16. Accuracy in relation to sentence length; AS_U: unlabeled attachment score, DA_U: unlabeled dependency accuracy, RA_U: root accuracy, EM_U: unlabeled exact match

Length	$\mathbf{AS}_U$	$\mathbf{DA}_U$	$\mathbf{RA}_U$	$\mathbf{EM}_U$
≤ 10	93.8	93.4	95.5	85.4
$11 \leq 20$	89.4	89.3	90.0	43.4
$21 \leq 30$	87.7	87.9	84.4	22.4
$31 \leq 40$	87.5	87.7	83.1	11.1
$41 \leq \infty$	86.8	87.1	73.7	5.3

a parsing methodology very similar to ours albeit with constituency-based representations, it is found that support vector machines give consistently higher accuracy than memory-based learning. We will return to this issue in the error analysis in the next section.

Turning finally to the assessment of labeled accuracy, we are not aware of any strictly comparable results, but Blaheta and Charniak (2000) report an F measure of 98.9% for the assignment of grammatical role labels to phrases that were correctly parsed by the parser described in Charniak (2000), using the same data set. If null labels are excluded, the F score drops to 95.6%. The corresponding F measures for our system, based on the labeled precision and recall for tokens that are assigned the correct head, are 98.0% and 97.8%, treating the default label DEP as the equivalent of a null label. The experiments are not strictly comparable, since they involve different sets of functional categories (where only the labels SBJ and PRD are equivalent) and one is based on phrase structure and the other on dependency structure, but it nevertheless seems fair to conclude that the labeling accuracy of our parser is close to the state of the art, even if its capacity to derive correct structures is not. It should also be kept in mind that the labeling here is performed in the same deterministic one-pass parsing process as the derivation of the structure, whereas Blaheta and Charniak (2000) apply an elaborate probabilistic model to the output of a probabilistic parser.

To make the comparison complete, it should also be pointed out that, if the memory-based deterministic approach dependency parsing does not quite reach the state of the art in terms of accuracy, it is highly competitive in terms of efficiency. This holds both with respect to parsing time, with a linear time parsing algorithm, and with respect to training time, where the memory-based approach vastly outperforms, e.g., support vector machines.

5.5.3 Error Analysis

Although we will not be able to present a detailed error analysis, we will try to tease apart some of the error sources involved in deterministic memory-based dependency parsing. More precisely, we will consider the influence of

Table 5.17. Accuracy as a function of tagging accuracy (English, model L_4'); AS: attachment score, EM: exact match; U: unlabeled, L: labeled

		AS		EM	
Tagging	**Accuracy**	**U**	**L**	**U**	**L**
Gold standard	100.0	89.7	88.3	36.0	31.8
Nakagawa et al. (2002)	97.1	88.7	87.1	34.4	29.8
Hall (2003)	96.1	88.1	86.3	32.8	28.4

part-of-speech tagging errors and the role of the inductively defined parser guide in relation to the deterministic parsing strategy.

Table 5.17 presents the parsing accuracy obtained with the best model for the English data, with three different ways of assigning part-of-speech tags in the preprocessing. The first uses the gold standard tags in the Penn Treebank annotation; the second uses the output of the tagger described in Nakagawa et al. (2002), based on revision learning with support vector machines and used in the parsing experiments of Yamada and Matsumoto (2003);[7] the third uses the output of the tagger described in Hall (2003), based on hidden Markov models with suffix probabilities and used in all the experiments of this study.

We see that there is a substantial improvement in parsing accuracy when using the gold standard tags, ranging from 1.6 percentage points for unlabeled attachment accuracy to 3.4 percentage points for labeled exact match. A more detailed look at different dependency types (not shown in the table) reveals that the greatest improvement is found for the argument types OBJ and PRD, where the labeled F measure increases by 4.1 (OBJ) and 3.2 (PRD) percentage points. This indicates that a significant proportion of the cases where these two types are confused during parsing are due to tagging errors. Using gold standard tags also improves the root accuracy of the parser from 86.4% to 90.0%, which is a very substantial difference.

Using the tagger of Nakagawa et al. (2002) instead of our own HMM tagger also makes a significant difference. In fact, although the error reduction in tagging is only about 25%, the error reduction in parsing is roughly 40% when compared to the scores obtained with gold standard tags, and as much as 50% for unlabeled exact match. It thus seems that the tagging errors avoided by the better tagger are of a kind that are important for syntactic parsing. However, without a detailed analysis of the differences between the taggers, it is difficult to say anything more precise than this.

The second kind of error analysis that we will present is an attempt to distinguish the influence of prediction errors, caused by errors in the learned approximation of the guide function, and errors that are the combined effect of previous prediction errors and the greedy, deterministic parsing strategy.

[7] Special thanks to Hiroyasu Yamada for supplying us with the output of this tagger for section 23 of the Wall Street Journal section of the Penn Treebank.

Table 5.18. Classification accuracy, number of instances, and number of exact matches (model L'_4); Swedish (section 0), English (section 23)

Data set	**Accuracy**	**Instances**	**Exact**
Swedish	90.7	16714	1064
English tagged	94.4	102419	15185
English gold	95.2	102419	15962

Table 5.18 presents the pure classification accuracy of the best memory-based models for learning and classification when tested on the set of instances derived from a gold standard parse of the respective test sets, section 0 of the Swedish treebank and section 23 of the English treebank. This proportion tells us how often the estimated guide prediction $\hat{g}(\Phi(c, A_x))$ coincides with the true oracle prediction $o(c, A_x)$ for the given data sets. The table also gives the number of instances in each test set (equivalent to the number of nondeterministic parser configurations in parsing the test set) and the number of parser states that had an exact match in the instance base.

First, we may note that the number of exact matches in the instance base is relatively low, about 6% for Swedish and about 15% for English. The higher proportion for English is expected, since the training data set is one order of magnitude larger. With a low percentage of exact matches it is crucial to have an adequate smoothing model, and it seems that the similarity-based smoothing built into the memory-based classification solves this problem very well, with classification accuracy above 90% for all conditions. For English we can also observe that tagging errors have a relatively small impact on classification accuracy as such, although the effect is magnified in parsing because of subsequent errors caused by the initial prediction error.

Comparing classification accuracy and parsing accuracy is not completely straightforward, but the most relevant metric is labeled attachment score, since the transitions chosen in the classification task are parameterized for dependency types. For Swedish, a classification accuracy of 90.7% corresponds to a labeled attachment score of 82.0%, which indicates a drop in accuracy of about 9% because of the greedy parsing strategy. For English with noisy tagging, a classification accuracy of 94.4% corresponds to a labeled attachment score of 86.3%, which is roughly 8% deterioration. For English with gold standard tags, there is only a 7% drop from 95.2% to 88.3%. As expected, the drop in accuracy from pure classification to parsing is smaller the better the classification accuracy, since a lower probability of prediction errors also leads to a lower probability of errors caused by earlier prediction errors. As a first approximation, we can predict the labeled attachment accuracy to be the square of the classification accuracy (CA) (treating both AS_L and CA as proportions ranging from 0 to 1):

$$\mathrm{AS}_L = \mathrm{CA}^2 \tag{5.5}$$

This prediction overestimates the parsing accuracy more for English than for Swedish. This can probably be explained by the fact that the English sentences are generally longer and more complex, which by itself increases the probability of errors being caused by previous prediction errors. In order to more accurately predict the parsing accuracy from the classification accuracy, we therefore need to introduce a constant c related to the complexity of the text samples being parsed:

$$\mathrm{AS}_L = c \cdot \mathrm{CA}^2 \tag{5.6}$$

In our experiments, the value of c appears to be approximately 0.97 for English and almost 1.0 for Swedish. Since the negative effect of error propagation is likely to increase with sentence length, we might be able to predict the value of c from the mean sentence length in the test corpus. For the test sets used here, a good approximation can be obtained by setting $c = 1 - 0.003(n - 15)$, where n is the mean number of words per sentence. Whether a model of this kind can be generalized to other languages and data sets is a question that can only be answered by further research.

In conclusion, it seems that the classification accuracy attained by the memory-based approach to learning is sufficient for highly accurate parsing, especially if sufficient amounts of training data are available, as indicated by the significantly higher accuracy for the English data sets. However, in order to convert this potential into state-of-the-art parsing accuracy, the parsing process clearly needs to be improved. This improvement may take the form of enhanced preprocessing, as suggested by the results of Isozaki et al. (2004), or it may require abandoning the strictly deterministic parsing strategy. We will return to these issues in the concluding chapter.