Journal of Machine Learning Research 2 (2002) 639–668 Submitted 9/01; Published 3/02

Shallow Parsing with PoS Taggers and Linguistic Features

Beáta Megyesi BEA@SPEECH.KTH.SE
Centre for Speech Technology (CTT)
Department of Speech, Music and Hearing
KTH, Sweden
Drottning Kristinas väg 31
SE–100 44, Stockholm, Sweden

Editors: James Hammerton, Miles Osborne, Susan Armstrong and Walter Daelemans

Abstract

Three data-driven publicly available part-of-speech taggers are applied to shallow parsing of Swedish texts. The phrase structure is represented by nine types of phrases in a hierarchical structure containing labels for every constituent type the token belongs to in the parse tree. The encoding is based on the concatenation of the phrase tags on the path from lowest to higher nodes. Various linguistic features are used in learning; the taggers are trained on the basis of lexical information only, part-of-speech only, and a combination of both, to predict the phrase structure of the tokens with or without part-of-speech. Special attention is directed to the taggers' sensitivity to different types of linguistic information included in learning, as well as the taggers' sensitivity to the size and the various types of training data sets. The method can be easily transferred to other languages.

Keywords: Chunking, Shallow parsing, Part-of-speech taggers, Hidden Markov models, Maximum entropy learning, Transformation-based learning

1. Introduction

Machine learning techniques in the last decade have permeated several areas of natural language processing (NLP). The reason is that a vast number of machine learning algorithms have proved to be able to learn from natural language data given a relatively small correctly annotated corpus. Therefore, machine learning algorithms make it possible to within a short period of time develop language resources—data analyzed on various linguistic levels—that are necessary for numerous applications in natural language processing.

One of the most popular NLP areas that machine learning algorithms have been successfully applied to is part-of-speech (PoS) tagging, i.e. the annotation of words with the contextually appropriate PoS tags, often including morphological features. The data-driven algorithms that have been successfully applied to this task for several languages include, among others, hidden Markov modeling (Brants, 2000), inductive logic programming (Cussens, 1998; Eineborg and Lindberg, 2000), maximum entropy learning (Ratnaparkhi, 1996), memory-based learning (Daelemans *et al.*, 1996; Zavrel and Daelemans, 1999), and transformation-based learning (Brill, 1994). The main advantage with data-driven PoS taggers is that they are language and tag set independent and thereby easily applicable to new languages and domains. The average accuracy that is reported for state-of-the-art data-driven PoS taggers lies between 95% and 98% depending on the language type the taggers are trained and tested on.

In the past years, some attempts also have been made to build data-driven shallow parsers. The main goal of the data-driven parsers is, above all, to find the phrase structure of the sentence and not, as one might think, to disambiguate words according to their context. The disambiguation is already taken care of by the PoS taggers which use some kind of background knowledge, i.e. parameters that tell the system to check the contextual environment of the current word and/or tag.

As a first step in building corpus-based parsers, a considerable amount of research has been carried out to find syntactically related non-overlapping groups of words, so-called "chunks" (Abney, 1991). A chunk is a major phrase category consisting of the phrasal head and its modifiers on the left hand side. The example below, borrowed from Tjong Kim Sang & Buchholz (2000), illustrates three different chunk types (NP, VP and PP) for the sentence "He reckons the current account deficit will narrow to only £1.8 billion in September" shown in bracketing structure.

> [NP He] [VP reckons] [NP the current account deficit] [VP will narrow]
> [PP to] [NP only £1.8 billion] [PP in] [NP September].

Within the area of data-driven chunking, much attention has been directed to the development of recognition methods for simple, non-recursive noun phrases, also called base NP chunks (e.g. Church, 1988; Cardie and Pierce, 1998; Skut and Brants, 1998). These phrases play an important role in many application areas, such as information extraction and information retrieval, as well as in human language processing (Gee and Grosjean, 1983). Research on the detection of other chunk types, such as prepositional phrases (PP), adverb phrases (ADVP), adjective phrases (ADJP) and verb clusters, by data-driven methods has also been carried out with promising results (see Ramshaw and Marcus, 1995; Argamon *et al.*, 1998; Brants, 1999; Buchholz *et al.*, 1999; Veenstra, 1999; Osborne, 2000; and Megyesi, 2001a). However, most of these chunkers only recognize a phrase up to its head word without finding the arguments on the right side of the head. For example, in the example above, the two PPs do not include their NP arguments. Additionally, in almost all these studies with the exception of work by Brants (1999), the internal phrase structure of the chunk is not analyzed. As we can see in the example sentence, the phrases inside the NP are not marked. Also, different studies use various linguistic information to find the chunks; some use PoS only without taking any lexical information under consideration, while some combine the words and their PoS in learning.

It is also worth mentioning that the majority of studies on chunking have been focused on the development of data-driven chunkers/parsers for English, just as was the case with the part of speech tagging task a couple of years ago. The reason is mainly that there is a correctly parsed corpus for English, the Penn Treebank (Marcus *et al.*, 1994), while such a corpus is missing for most languages. Given this correctly parsed large data set, the development and evaluation of data-driven approaches become easier and more reliable.

The motivating purpose of this work is to build a data-driven shallow parser without a great deal of human effort for Swedish, describing the whole constituent structure the word belongs to in a hierarchical fashion. Desirable properties of the shallow parser are as follows:

- easily trainable, fast and robust
- corpus-based, i.e. data-driven, so that it can be applicable to various domains
- having a hierarchical phrase representation so that it is capable of being used for many different applications

The fact that many data-driven PoS taggers are language and tag set independent, and the fact that these taggers have some implemented linguistic knowledge about the contextual environment of words and/or tags, lead to the thought that these PoS taggers can be assumed to be useful to parse texts, given some correctly chunked/parsed data, i.e. a treebank. Inspired by the success of the maximum entropy based data-driven PoS tagger, mxpost (Ratnaparkhi, 1996), applied directly to chunk English (Osborne, 2000), we will use three different data-driven PoS taggers as a basis for parsing Swedish texts. The PoS taggers are implementations of three algorithms: hidden Markov modeling (Brants, 2000), maximum entropy learning (Ratnaparkhi, 1996), and transformation-based learning (Brill, 1994).

The aim of this study is, in particular, to find out what combinations of linguistic information are the most appropriate for the parsing task so that the taggers can efficiently learn to parse texts, and to find out what effects different kinds of linguistic information included in the training data has on the parsers in this processing. In addition, the taggers' sensitivity to the size of the training set is investigated, and an evaluation for real-world applications is carried out.

The remainder of the paper is organized as follows: Section 2 gives an overview of previous studies performed on data-driven chunking; Section 3 presents the phrase structure representation, the training data and benchmark, as well as a brief description of the taggers that the parsers are built on; Section 4 describes the experiments on various linguistic features used in learning; Section 5 presents the experiments and the results; and finally, Section 6 concludes the paper and gives directions for further research.

2. Previous Work on Data-Driven Text Chunking

The concept of the chunk was introduced by Abney (1991). He defines a chunk as "a single content word surrounded by a constellation of function words, matching a fixed template". He proposed that by dividing a sentence into meaningful, correlated sequences of words—chunks—and combining those into trees, we can build a parser which has psycholinguistic evidence in that it represents structures corresponding to pauses and intonation changes in speech. Abney's chunk parser consists of two steps; first the chunker finds potential chunks on the basis of PoS information, and then an attacher finds the correct chunk by resolving structural ambiguities on the basis of lexical information.

Abney's pioneering work has influenced a lot of researchers. Several studies have been performed to develop data-driven chunkers as a first step to build parsers. One of the earliest studies on this topic was presented by Ramshaw and Marcus (1995). They used transformation-based learning (Brill, 1994) to locate chunks in texts by treating chunking as a tagging problem. The chunk structure was represented as tags attached to words, in a similar way as is done in data-driven PoS tagging. They performed experiments using two different chunk structure targets. The first target was to identify non-overlapping, non-recursive noun phrases, so called base NPs, as far as the nominal head, including determiners and adjectives, but not prepositional phrases or other types of arguments located after the head word. The tag set consisted of three types of tags: B for the first word of the base NP, I for the words inside the base NP, and O for the words outside of the chunk. The second target of their work was to partition sentences into non-overlapping noun-type (N) and verb-type (V) chunks in a similar fashion as was proposed by Abney (1991). The noun-type chunks consisted of, among others, noun phrases as far as the nominal head, prepositional phrases including an NP argument, but not coordinating conjoined NPs. Each N and V type had two tags, depending on whether the word was initially positioned in the type or not, and an extra tag was reserved for punctuation marks. They used the parsed Wall Street Journal texts from Penn Treebank (Marcus *et al.*, 1994) to automatically derive the chunk structure. They extended the templates of Brill's PoS tagger to include references up to two chunk tags, as well as to up to three words and/or their PoS tags. The result showed a precision of 93.1% and a recall of 93.5% for base NP chunks when trained on 950k words and tested on 50k words using lexical and PoS information. When lexical information was excluded, precision and recall decreased to 90.5% and 90.7% respectively. For the N and V partitioning, precision and recall rates are reported to be 88% when training was performed on 200k words. Also, they pointed out that the size of the training set has a significant effect on the results.

Argamon *et al.* (1998) used memory-based sequence learning to recognize NP and VP chunks in PoS tagged texts. The same data set was used as in the study by Ramshaw and Marcus (1995) but the learner was trained on PoS tag sequences containing bracketed chunk boundaries without including lexical information. They report precision and recall rates of 91.6%.

Other experiments on data-driven chunking were also performed with memory-based learning methods. Cardie and Pierce (1998) presented a corpus-based approach for finding base NPs by using PoS tag sequences without any lexical information. They created grammar rules from the training data and improved the grammar by pruning it on another data set, using local repair heuristics that improved the precision without decreasing the recall. A further step of discarding the ten worst rules was also carried out without decreasing the precision. They achieved 94% precision and recall on simple base NPs, and 91% on more complex ones.

Veenstra (1999), also using a memory-based learning technique—IGTree—(Daelemans *et al.*, 1996), described experiments on NP, VP and PP chunking using the Wall Street Journal for data and the BIO labels attached to each chunk type as it was proposed by Ramshaw and Marcus (1995) and described above. He reported precision and recall rates between 94% – 95%, and accuracy of 98% and 99% for NP and VP chunks respectively.

Buchholz *et al.* (1999) used memory-based learning to assign grammatical relations (for example subject, object, etc.) to texts by first finding NP, VP, PP, ADJP and ADVP chunks, and then using pairs of chunks to predict grammatical relations. The data-driven chunker was in turn applied in several steps. First, prepositions, NP, VP, ADJP and ADVP chunks were found simultaneously, then prepositions and NPs were collapsed into PPs. They reported $F_{\beta=1}$ score of 92.3% for NPs, 91.8% for VPs, 66.7% for AP chunks, 77.9% for ADVP chunks, and 96.1% for prepositions. For PP chunks, the $F_{\beta=1}$ score was 92%.

Brants (1999) presented a method for partial parsing that uses cascades of Markov Models to generate structural elements in a layer-by-layer fashion. The algorithm generates the internal structure of NP and PP chunks including APs and ADVPs, and other pre-modifiers. Sequences of words divided sentence by sentence served as input and the output was the PoS and chunked text. The algorithm was tested on 300k words taken from the NEGRA corpus consisting of German newspaper texts. Recall was 54% for 1 layer and 84.8% for 9 layers; precision was 91.4% for 1 layer and 88.3% for 9 layers. As Brants points out, these results are not directly comparable to previous studies because his study was performed on a different language than English (namely German) and his algorithm labeled the internal phrases within the NP and PP chunks.

Osborne (2000) used a maximum entropy-based PoS tagger, MXPOST (Ratnaparkhi, 1996), without modifying the PoS tagger's internal operation, thus treating chunking as part-of-speech tagging, with an accuracy of 94.88% and an overall $F_{\beta=1}$ score of 91.94%. The study was a part of a competition for the chunking approach at the 4th Conference on Computational Natural Language Learning (CoNLL-2000) which supplied the tag set, including the training and test data taken from the Wall Street Journal corpus. The training data consisted of 211,727 tokens and the test data of 47,377 tokens. The types of chunks used in the competition are described by Tjong Kim Sang & Buchholz (2000) and include "base phrase categories": noun phrases (NP) to the nominal head, verb clusters (VP), adjective phrases (ADJP), adverb phrases (ADVP), prepositions (PP) without NPs, compound conjunctions, verbal particles, interjections, list markers and conjunctions.

The goal of the studies presented above was mainly to identify base phrase categories. Next, we will describe our method to build data-driven shallow parsers representing general phrasing including, among others, whole noun phrases with right-side arguments.

3. Building Shallow Parsers

Four different aspects need to be addressed in order to build a data-driven shallow parser; the choice and the representation of the target classes that the algorithms have to learn to predict, the data used for training and test, the choice of algorithm(s), and the attributes or features included in learning. In the following sections, these aspects will be described.

3.1 Phrase Structure Representation

As we have seen in Section 2, in previous studies (with the exception of the work presented by Brants, 1999), the internal structure of the chunks is not analyzed. Only categories on higher nodes of the constituent structure are represented. For example, if a token/word belongs to an adjective phrase which in turn belongs to a noun phrase, the token is labeled with the noun phrase constituent only, not marking any other lower nodes in the tree. Leaving out the lowest constituents the token belongs to can have drawbacks for several applications, for example in dialog systems or text-to-speech systems, where information about the whole constituent structure can be important for better system performance. Therefore, the representation of the whole phrasal hierarchy containing information on all phrases is desirable.

Additionally, previous studies represent only partially linguistically motivated phrasal categories. Some phrase structures are not fully represented. For example, noun phrases are marked as far as the head noun only, hence the arguments on the right side of the noun head are missing. Also, prepositional phrases in many studies do not include any noun phrase. Furthermore, some PoS categories are treated as phrases, as in the CoNLL-2000 competition on chunking, where conjunctions constitute a conjunction phrase and interjections an interjection phrase.

To be able to represent the whole hierarchical phrase structure, nine types of phrases are used. Some categories correspond to the chunks used in previous studies, for example AP, ADVP, and verb clusters. Other categories are designed to be able to handle arguments on the right hand side of the phrasal head and represent maximal projections, such as the maximal noun phrase label. Some categories are included to handle co-ordinated phrases, such as the maximal adjective phrase label. The phrase categories are listed below, each followed by a brief explanation and an example.

- Adverb Phrase (ADVP) consists of adverbs that can modify adjectives or numerical expressions.
  e.g. very

- Minimal Adjective Phrase (AP) constitutes the adjectival head and its possible modifiers, e.g. ADVP and/or prepositional phrase.
  e.g. very interesting

- Maximal Adjective Phrase (APMAX) includes more than one AP with a delimiter or a conjunction in between.
  e.g. very interesting and nice

- Numerical Expression (NUMP) consists of numerals with their possible modifiers, for example AP or ADVP.
  e.g. several thousands

- Noun Phrase (NP) may include the head noun and its modifiers to the left, e.g. determiners, nouns in genitive, possessive pronouns, numerical expressions, AP, APMAX and/or compound nouns. Thus, possessive expressions do not split an NP into two noun phrases as in the CoNLL-2000 shared task on chunking.
  e.g. Pilger's very interesting and nice book

- Maximal Projection of NP (NPMAX) includes one or more NP(s) with following PP(s) as possible modifier.
  e.g. Pilger's very interesting and nice book about politics

- Prepositional Phrase (PP) consists of one or several prepositions delimited by a conjunction and one or several NPs/NPMAXs, or in elliptical expressions an AP only.
  e.g. about politics

- Verb Cluster (VC) consists of a continuous verb group belonging to the same verb phrase without any intervening constituents like NP or ADVP.
  e.g. would have been

- Infinitive Phrase (INFP) includes an infinite verb together with the infinite particle and may contain ADVP and/or verbal particles.
  e.g. to go out

Note that the grammatical categories represent neither clauses, such as relative clauses, nor sentences. These structures are planned to be analyzed in a later stage.

3.2 Training Data and Benchmark

Swedish belongs to the Scandinavian, North Germanic family of the Germanic branch of Indo-European languages. It is morphologically richer than, for example, English. Nouns in general have a two-gender distinction. The genders are marked mainly by articles, adjectives, anaphoric pronouns, and in plural endings. As in English, nouns can appear with or without articles. There are, however, definite and indefinite articles that agree with the head noun in gender, number and definiteness. Furthermore, adjectives have gender, definiteness and plurality markers. Thus, in a noun phrase, both articles and adjectives agree in number, gender and definiteness with the head noun. Also, compound nouns are frequent and productive. Verbs lack markers for person or number of the subject but retain tense including complex tense forms. From a syntactic point of view, Swedish has subject-verb-object order in independent declarative sentences, as well as in subordinate clauses, similar to English. However, in subordinate clauses the sentence adverbs normally precede the finite verb and the perfect auxiliary can be omitted.

Unfortunately, correctly chunked/parsed texts are not available for Swedish. Therefore, a treebank was built to serve as training data and a benchmark corpus. For the treebank development, an Earley Parser, SPARK (Aycock, 1998) was used together with a context-free grammar for Swedish developed by the author.

The second version of the Stockholm-Umeå corpus (Ejerhed *et al.*, 1992) annotated with PAROLE tags served as input to the parser.[1] The corpus is balanced, consisting of over one million PoS tagged tokens taken from different text genres in Swedish. The tag set consists of 146 tags including PoS categories and morphological features. The PoS tagged texts were parsed by SPARK using the nine phrase categories that were described in Section 3.1.

Each phrase type is represented with an additional tag marking position information in a manner similar to that proposed by Ramshaw and Marcus (1995) and used in the CoNLL-2000 competition:

XB – the initial word of the phrase X
XI – non-initial word inside the phrase X
O – word outside of any phrase.

Thus, each word and punctuation mark in a sentence is accompanied by a tag which indicates the phrase structure the token belongs to in the parse tree together with the position information. Since a token may belong to several phrases, it can have several tags.

The representation is illustrated in the example below for the Swedish equivalent of the sentence "Everybody should read Pilger's very good books about politics" represented first by parenthesis notation, and second by PoS and phrase tags.

1. Thanks to Britt Hartmann at the Department of Linguistics, Stockholm University, Sweden for making the second version of the Stockholm-Umeå corpus with PAROLE tags available.

[NP Alla NP] [VC borde läsa VC] [NPMAX [NP Pilgers [AP [ADVP mycket ADVP] bra AP] böcker NP] [PP om [NP politik NP] PP] NPMAX].

Word	PoS + morphology as PAROLE tags	Phrase tags	Translation
Alla	PI@0P0@S	NPB	(everybody)
borde	V@IIAS	VCB	(should)
läsa	V@N0AS	VCI	(read)
Pilgers	NP00G@0S	NPB_NPMAXB	(Pilger's)
mycket	RGPS	ADVPB_APB_NPI_NPMAXI	(very)
bra	AQP00N0S	API_NPI_NPMAXI	(good)
böcker	NCUPN@IS	NPI_NPMAXI	(books)
om	SPS	PPB_NPMAXI	(about)
politik	NCUSN@IS	NPB_PPI_NPMAXI	(politics)
.	FE	0	

The label for a word forms a hierarchical grouping of the parts of the sentence into constituents where lower nodes are situated nearest the word and higher nodes are farthest out. The advantage of the hierarchical annotation on the phrase level is that the user can choose the level of analysis by skipping phrase categories on lower, or higher nodes. For example, the user may only want to use noun phrase extraction without any information on the constituents inside the noun phrase, or to get a full analysis of every large phrase in the sentence. This type of annotation can be used in many different applications. The question is how well the data-driven PoS taggers can learn the hierarchical phrasal structure.

The parsed text, annotated with the hierarchical constituent structure serves as training data and benchmark corpus for the experiments. SPARK introduced some errors in both the training and benchmark. The error rate is estimated between 6% and 11% with 98% confidence, and was determined by calculating the errors on a sample of 2,450 tokens in the training and test sets respectively. 60% of the errors are due to PP attachment problem in maximal projections of NPs, which is considered to be difficult even for human annotators. About 25% of the noise is due to wrong position information of the NP. The rest of the errors can be found mainly in connection to adjective phrases. As manual post-processing to eliminate the noise was found to be prohibitively time-consuming, these errors have not been corrected.

After this description of the representation of the data, a brief overview of the algorithms, each with implementations for the PoS tagging approach that the parsers are built on, follows.

3.3 Algorithms and Implementations

The shallow parsers are based on three state-of-the-art data-driven algorithms that have implementations for the PoS tagging approach. Common to these taggers are their language and tag set independence, their free availability for research and their successful usage for several languages. The taggers that will be used to parse Swedish in this study are: fntbl (Ngai and Florian, 2001) which is a fast version of Brill's tagger based on transformation-based learning (Brill, 1994), mxpost, based on the maximum entropy framework (Ratnaparkhi, 1996), and lastly, Trigrams'n'Tags (tnt) based on a Hidden Markov Model (Brants, 2000).

fntbl, developed by Ngai and Florian (2001), is a fast version of Brill's transformation-based learning algorithm[2]. It is a rule-based approach that learns by detecting errors. It begins with an unannotated text that is labeled by an initial-state annotator in a heuristic fashion. Known words (according to some lexicon) are annotated with their most frequent tag while unknown words receive an initial tag (for example the most frequently occurring tag in the corpus). Then, an ordered list of rules learned during training is applied deterministically to change the tags of the words according to their contexts. Unknown words are first assumed to be nouns and handled by prefix and suffix analysis by looking at the first/last one to four letters, capitalization feature and adjacent word co-occurrence. For the disambiguation of known words, the system uses a context of up to three preceding and following words and/or tags of the focus word as default.

mxpost, developed by Ratnaparkhi (1996), is a probabilistic classification-based approach based on a maximum entropy model where contextual information is represented as binary features that are used simultaneously in order to predict the PoS tags. The binary features used by mxpost as default include the current word, the following and preceding two words and the preceding one or two tags. For rare and unknown words the first and last four characters are included in the features, as well as information about whether the word contains uppercase characters, hyphens or numbers. The tagger uses a beam search in order to find the most probable sequence of tags. The tag sequence with the highest probability is chosen.

Trigrams'n'Tags (tnt), developed by Brants (2000), is a statistical approach, based on a hidden Markov model that uses the Viterbi algorithm with beam search for fast processing. The states represent tags and the transition probabilities depend on pairs of tags. The system uses maximum likelihood probabilities derived from the relative frequencies. The main smoothing technique implemented as default is linear interpolation. The system uses a context of three tags. Unknown words are handled by suffix analysis up to the last ten characters of the word. Additionally, information about capitalization is included as default.

For the experiments, all systems are used with the default settings according to their documentation and were trained on the Swedish data described in Section 3.2.

4. Experiments on Various Linguistic Features in Learning

In previous studies on chunking, different types of linguistic information was used in training in order to find the correct chunk structure of the sentence. Ramshaw and Marcus (1995) used lexical and/or PoS information, Argamon *et al.* (1998) and Cardie and Pierce (1998)

2. The difference between Brill's original and Ngai & Florian's implementation (fntbl) is that the latter stores the rules in memory instead of regenerating the rules at each step of the learning process, and the rules are only generated for the examples that change. A detailed description can be found in Ngai and Florian (2001).

induced learning on the basis of PoS sequences without including any lexical information, while Brants (1999) entirely relied on words to be able to recognize both the PoS tags and the chunks. Comparing the results of these studies, we can see that the average accuracy is reported to be lowest when training is performed on PoS sequences only. However, it is difficult to compare the results because either the learning algorithm, the data set or the language vary across the studies. Therefore, it is of particular interest to train, test and compare the taggers on different types of data sets containing various linguistic features, using the same training and test set for reliable evaluation.

In order to ascertain how well different data-driven PoS taggers can learn the whole hierarchical constituent structure of the word sequences, and to examine what effect different kinds of linguistic information included in the training data have on the taggers, four experiments are carried out. Each tagger is trained on four types of data set, each including different types of linguistic information, as is shown in Table 1. First, the training is performed on the basis of the word only—lexical information—to predict the PoS tag and the phrase tags. Second, similarly to the first case, training is performed on the word sequences to predict the phrase tags without PoS information. Third, the training is based on the word together with its PoS to predict the phrase labels. Lastly, the words are removed from the training data, and only the PoS tags of the words are trained with phrase labels. In this way, all combinations of possible types (word and/or PoS) and possible target classes (phrases with or without PoS) are examined.

types to learn from	target classes
Words	PoS + Phrases
Words	Phrases
Words + PoS	Phrases
PoS	Phrases

Table 1: Combinations of the linguistic features in learning.

In order to examine how the size of the training set influences the performance of the classifiers, each tagger is trained in each experiment nine times on the subsets of same data set of various sizes from one thousand to five hundred thousand tokens: 1k, 2k, 5k, 10k, 20k, 50k, 100k, 200k, and 500k tokens respectively. Then, the same test set, consisting of 117,530 tokens, is annotated by each classifier. In each experiment, the training and test sets are disjoint.

5. Results

In this section, the results from the four learning tasks as described in Section 4 are presented. In each experiment, the evaluation is based on the widely used measure *accuracy*, which is obtained by dividing the number of correctly labeled tokens with the total number of tokens, see Equation 1. A correct parse requires complete and correct phrase labels for a token including the position information (bio tag). If the word would lack a label for a phrase that it is part of, or if a phrase label would have wrong position information then the whole tag is considered to be incorrect.

$$Accuracy = \frac{\#\, of\, correctly\, tagged\, tokens}{\#\, of\, tokens} \tag{1}$$

In some cases, the performance of the classifiers is also measured with precision, recall, and $F_{\beta=1}$ rates for each single phrase type given by the hierarchical annotation. Each phrase type is extracted from the concatenated phrase label and counted as described below in Equations 2, 3 and 4.

$$Precision = \frac{\#\, of\, correctly\, tagged\, tokens\, as\, phrase\, type\, X}{\#\, of\, detected\, tokens\, as\, phrase\, type\, X} \tag{2}$$

$$Recall = \frac{\#\, of\, correctly\, tagged\, tokens\, as\, phrase\, type\, X}{\#\, of\, tokens\, as\, phrase\, type\, X} \tag{3}$$

$$F_{\beta=1} = \frac{(\beta^2 + 1) * Precision * Recall}{\beta^2 * Precision + Recall} \tag{4}$$

Before we go into details about the results, several aspects that might influence the performance of the classifiers have to be considered. One of these concerns the number of target classes the learners learn to predict in the different experiments when training is performed on various sizes of data sets with different linguistic features involved in learning. Due to the the hierarchical annotation, the number of possible combinations of phrase types lies between 260 and 3100 classes, depending on the size and the type of the training data. The relationship between the size of the training set and the number of classes that the learners search through to predict the phrase tags with and without PoS information is shown in Figure 1. We can see that the number of target classes increases with the size of the training set, as well as when the prediction of PoS tags together with phrase tags are required by the learners.

Another aspect that might have an influence on the performance concerns the number of token types appearing in the training data, i.e. the number of different lexical token types, part-of-speech tags or a combination of these. Figure 2 shows, not surprisingly, that the number of token types increases with the size of the training set. The increase is largest when lexical information serves as a basis in the learning process, and lowest when training is performed on PoS sequences only without the presence of the words due to the low number of PoS tags; The total number of PoS tags lies between 82 and 143 depending on the size of the training set. It is also worth noting that the number of types is somewhat higher when both lexical and PoS information is included in the training to learn the phrase categories, compared to when only the words are present. The reason is, naturally, that there are no homonyms because of the presence of the PoS tags attached to each token.

The percentage of token types that can have more than one target class in the training data is also of interest since the algorithms have to learn to choose the correct class among the possible ones given a certain context. Figure 3 shows the percentage of ambiguous token types in the training data of various size for the four types of learning experiments. Between

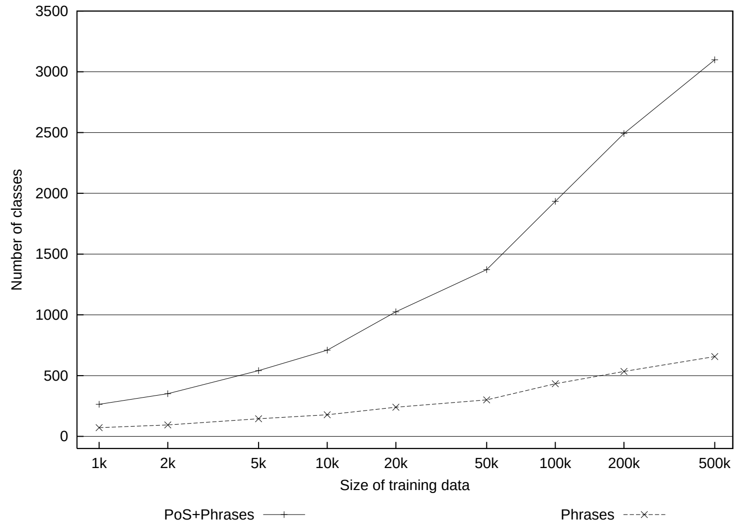

Figure 1: The number of target classes in training data of various size.

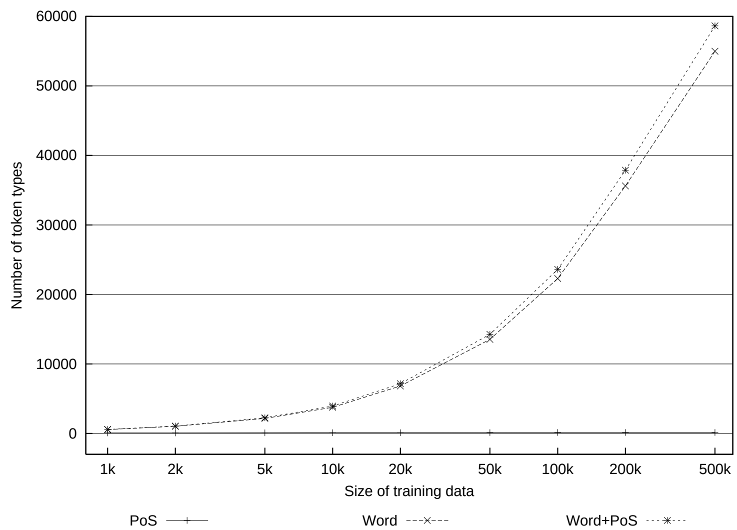

Figure 2: The number of token types in training data of various size.

60% and 85% of the PoS types are ambiguous while for the lexical types including words with or without PoS the percentage of ambiguous types is significantly lower. However, when the target class constitutes PoS and phrase structure, the number of ambiguous token types is higher than it is when the target class contains phrase labels alone.

The percentage of unknown tokens is also of interest since the classification task becomes harder when the test set includes a large number of unknown tokens, tokens that are not present in the training data. Figure 4 illustrates the percentage of unknown tokens in the test set compared to the training sets of different sizes for the various learning tasks. Not surprisingly, the number of unknown tokens is very small or zero when training is performed

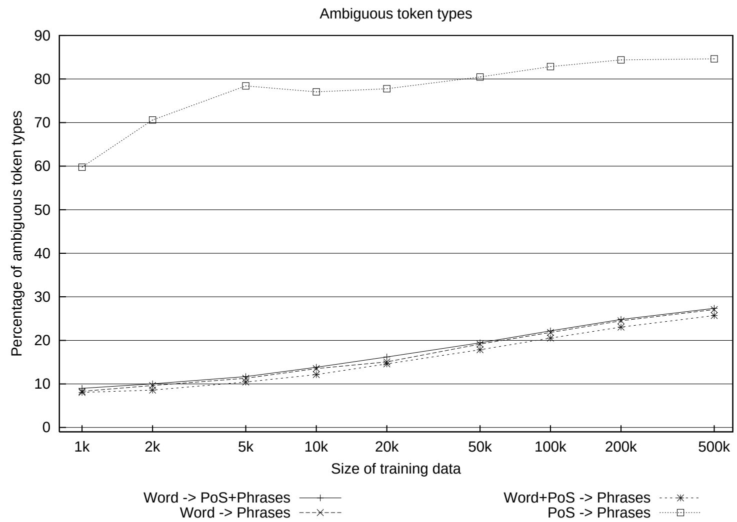

Figure 3: The percentage of ambiguous token types in training data of various size.

on PoS sequences only since the majority of PoS tags appears in the training data. The largest number of unknown tokens is found when learning is based on lexical and PoS information on smaller training corpora containing up to 100k tokens. On the other hand, if large training set is used, containing both lexical and PoS information, the number of unknown words decreases compared to when training is performed on lexical information only.

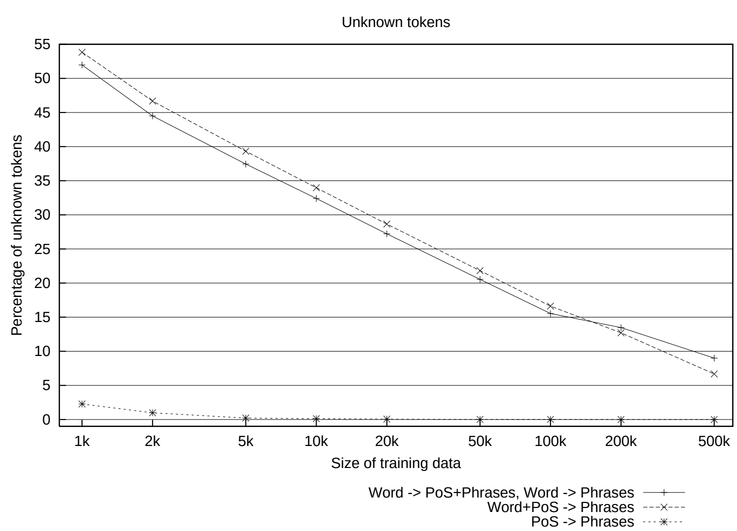

Figure 4: The percentage of unknown tokens in the test data compared to the training data of various size.

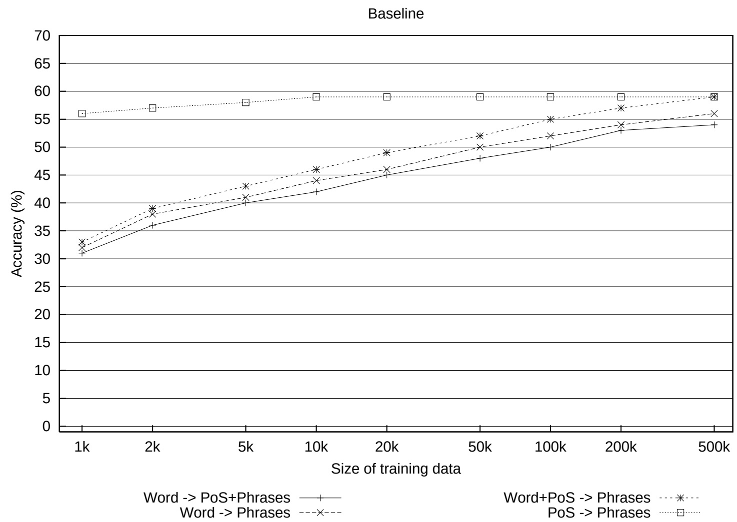

Figure 5: Baseline performance for each experiment for training data of various sizes.

Lastly, in order to evaluate the effectiveness of the classifiers for the four learning tasks, baseline performance is relevant since it describes a minimal performance rate that each classifier should achieve. Baseline values have therefore been obtained for the test data of the four types of learning tasks. The baseline is counted in different ways depending on the input the learners get and the class they have to learn to predict. Each known token in the test data receives a class label (i.e. either PoS + Phrases, or Phrases) that is most frequently associated with that token type in the training data. Tokens not in the training data are treated as wrongly annotated. In Figure 5, the results are shown for the training data of various sizes within each experiment type. On average, baseline performance is lowest when lexical information is involved in training. When PoS categories are also included in the training set, baseline performance increases. We can also notice that the size of the training set influences the accuracy; when training is performed on large training corpora, the baseline accuracies for the four types of training sets become more even.

With these prerequisites in mind, the results given by the classifiers for each learning task will be described.

5.1 Performance of the Classifiers

To present an overall picture of the parsers' performance, the accuracy of each classifier, when training is performed on 200k tokens, is listed in Table 2. Performance measures for known and unknown tokens are also listed separately. The performance of the classifiers varies depending on what type of information is included in the training data. The best average performance of all three parsers is achieved when only PoS information constitutes the input to the classifiers. When PoS information is not present in learning, the accuracy of all algorithms drops markedly.

The transformation-based learner, fntbl, achieves best performance when only PoS information is included in training, while in the other experiments, the maximum entropy

TYPES	number	CLASSES	number	FNTBL			MXPOST			TNT		
				T	K	U	T	K	U	T	K	U
Words	35,611	PoS + Phrases	2,492	72.8	77.6	36.9	**77.9**	*80.1*	*61.2*	72.2	75.0	51.9
Words	35,611	Phrases	534	75.1	77.9	54.4	**81.7**	*83.0*	*72.0*	72.8	75.0	56.3
Words + PoS	37,870	Phrases	534	83.3	83.5	82.4	**87.9**	*88.4*	*84.3*	79.9	80.0	79.5
PoS	141	Phrases	534	**94.8**	*94.8*	*70.4*	90.0	90.0	20.0	92.0	92.1	40.0

Table 2: The results are given for each classifier when trained on 200k tokens on the four types of input, and tested on 117,530 tokens. Accuracy (%) is calculated for the total number of tokens (T), as well as for known (K) and unknown (U) tokens.

tagger, MXPOST, obtains highest accuracy. However, when training is performed on lexical sequences only, TNT obtains better results for the annotation of unknown tokens, than FNTBL does.

The nine phrase types are also evaluated separately by extracting each phrase type from the concatenated tags, i.e. by not considering the correctness of the phrasal categories on lower and/or higher nodes in the tree in the evaluation process. Precision, recall and $F_{\beta=1}$ rates are measured for the phrase types given by the parsers trained on 200k tokens on various types of input features, and tested on 117,530 tokens. The $F_{\beta=1}$ scores are given in Table 3, and the complete set of values are listed in the Appendix in Table 8. Highest scores for a phrase type are printed in bold, while the highest values for the various learning types are italicized.

Feature	Class	ADVP	AP	APMAX	INFP	NP	NPMAX	NUMP	PP	VC
	Total	5,970	10,477	1,433	2,541	53,810	24,350	1,951	29,419	16,282
$W \rightarrow PoS + Ph$	FNTBL	*83.5*	74.0	14.4	87.4	94.2	77.3	90.1	85.0	93.5
	MXPOST	81.2	78.3	24.8	*93.8*	95.7	*79.4*	89.5	*88.4*	96.1
	TnT	*83.5*	*80.3*	*27.1*	88.9	*96.3*	66.8	*91.9*	79.3	*96.8*
$W \rightarrow Ph$	FNTBL	*83.0*	74.3	22.5	86.4	94.2	74.4	91.0	83.0	93.5
	MXPOST	80.9	79.1	*28.8*	*89.9*	*95.5*	*80.0*	89.7	*88.7*	95.7
	TnT	81.6	*79.2*	27.9	87.3	95.5	66.3	*91.5*	78.3	*96.0*
$W{+}PoS \rightarrow Ph$	FNTBL	99.3	86.0	41.2	93.9	97.4	81.0	95.6	87.9	99.2
	MXPOST	98.5	89.1	*47.2*	*97.3*	*98.3*	*84.6*	94.3	*90.9*	*99.4*
	TnT	**99.4**	*89.3*	43.0	91.8	98.0	73.1	*96.7*	83.0	99.1
$PoS \rightarrow Ph$	FNTBL	*80.5*	**95.3**	**86.6**	**100.0**	**99.3**	**97.6**	98.0	**98.1**	**100.0**
	MXPOST	77.7	91.9	75.1	99.0	98.7	87.2	96.9	93.3	99.9
	TnT	76.8	93.9	78.0	98.3	98.9	95.7	**98.4**	96.7	99.8

Table 3: $F_{\beta=1}$ rates for each classifier when trained on 200k tokens on the four types of input features, and tested on 117,530 tokens. The total number of occurrences for each phrase type in the benchmark is given in the second row.

On average, verb clusters (vc) and infinitive phrases (infp) are easiest to classify, followed by noun phrases (np), prepositional phrases (pp), and numerical expressions (nump). Adjective phrases, especially the maximal projections of aps (apmax), receive a surprisingly low $F_{\beta=1}$, when lexical information is involved in the learning task. Most of the conjoined aps are not found by the classifiers at all—the recall values are exceptionally low—as shown in Table 8. Maximal projections of noun phrases (npmax) are also difficult to detect compared to other phrase types, even though the recall rates are considerably higher than for apmax. The low recall values for the maximal projections in general could be the result of the biased training data and benchmark caused by the rule-based context-free parser.

We can see that the best values for eight of the phrases are achieved when training is performed on PoS sequences only. Adverb phrases (advp), on the other hand, are more often correctly detected when lexical information is included in learning. An explanation for this can be found in the annotation of adverbs in the SUC corpus where the discrimination of the adverbs is made on the basis of their morphological structure, rather than their syntactic characteristics. Thus, sentence adverbs do not belong to a distinct PoS category. For this reason these adverbs had to be listed in the rule-based parser in order to correctly detect the phrase structure. Therefore, the data-driven parsers, when trained on PoS sequences only, wrongly analyze adverb phrases, shown by the comparatively low precision.

These results do not tell us about the algorithms' sensitivity to the size of the training set when different types of information are used in learning. One might surmise that the larger amount of data we use, the better performance we get. However, the improvement does not necessarily have to be the same for the algorithms when we train them on various input features. Next, the effect of the different linguistic information used in learning will be described.

5.2 The Effect of the Linguistic Features

The results for each experiment on the learners' sensitivity to the input feature sets (word, word and PoS, PoS only), and to the number of target classes (phrases with or without PoS), are shown in Figures 6, 7, 8 and 9, respectively. All systems in all four experiments outperform the baseline independently of the type of linguistic features involved in learning or of the size of the training set.

The first learning task, where training is performed on the basis of lexical information only, to predict the PoS together with the correct phrase labels (word → PoS + phrases), is the most difficult classification task for every algorithm (see Figure 6). This is not surprising since the systems have to learn a great number of classes, between 264 and 3099 tags, depending on the size of the training set. Thus, in this experiment, the hypothesis space that the algorithms have to search through is large. The classifiers here are treated as PoS taggers and parsers. tnt has the lowest error rate when training is performed on small training sets consisting of up to 20k tokens while mxpost outperforms tnt when using 50k or more tokens for training. It is also worth noticing that fntbl achieves higher performance than mxpost when training is done on very small training sets because of fntbl's higher accuracy achieved for the annotation of known words.

In the second learning task, where PoS information is not present in the training data, i.e. the training is performed entirely on lexical information (word → phrases), the hypothesis

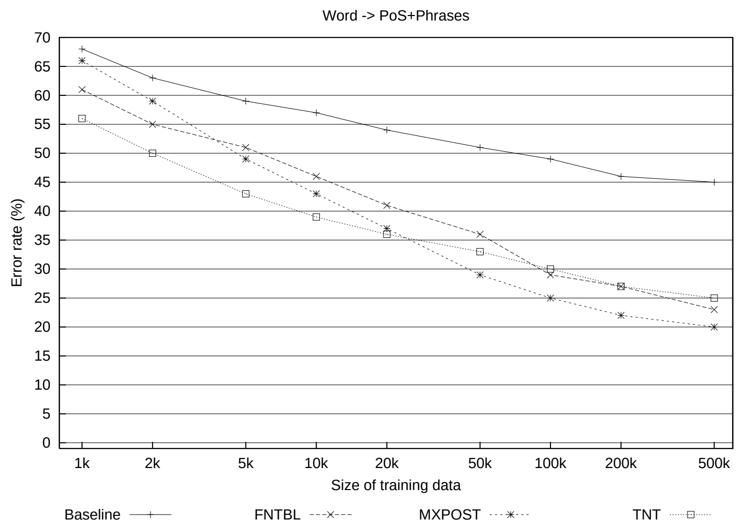

Figure 6: The error rate for each classifier when training is performed on the basis of lexical information to classify PoS and phrase structure information.

space becomes smaller than in the first experiment due to a decrease in the number of classes. The smaller tag set makes the classification task easier and average system performance increases (see Figure 7). Similarly to the first experiment, the maximum entropy approach, MXPOST, achieves the lowest error rate in cases in which the training corpus consists of more than 5k tokens. TNT obtains the best result when the training set is small (up to 5k tokens), while FNTBL outperforms TNT on large training sets (200k tokens or more).

In the third learning task, where both lexical and PoS information is included as input features for the recognition of the phrasal structures (WORD + PoS → PHRASES), the average performance of the classifiers further increases (see Figure 8). A possible explanation for the increase of the systems' performance can be that although in this experiment we find the largest number of token types, the problem of lexical homonymy is eliminated, since every token type becomes unique by the PoS tag attached to it. We thereby reduce the number of possible parse trees. Just as in the first two experiments, MXPOST has the lowest error rate when a large training set is used in learning (5k tokens or more), and TNT succeeds well when learning is performed on small data sets (up to 5k tokens). FNTBL succeeds better than TNT when large training sets serve as input (200k tokens or more), and has highest error rate when training is carried out on small corpora (up to 50k tokens).

Lastly, in the fourth learning task, where lexical information is not present in training (PoS → PHRASES), the performance of the systems increases greatly compared to cases in which lexical information is included in the training process. This can be explained by the low percentage of unknown tokens (PoS tags) in small training sets, and the absence of unknown tokens when a large training corpus is used (see Figure 4). The baseline performance therefore increases and the learning curves of the classifiers converge (see Figure 9). FNTBL obtains the best accuracy on average compared to the statistical approaches TNT and MXPOST.

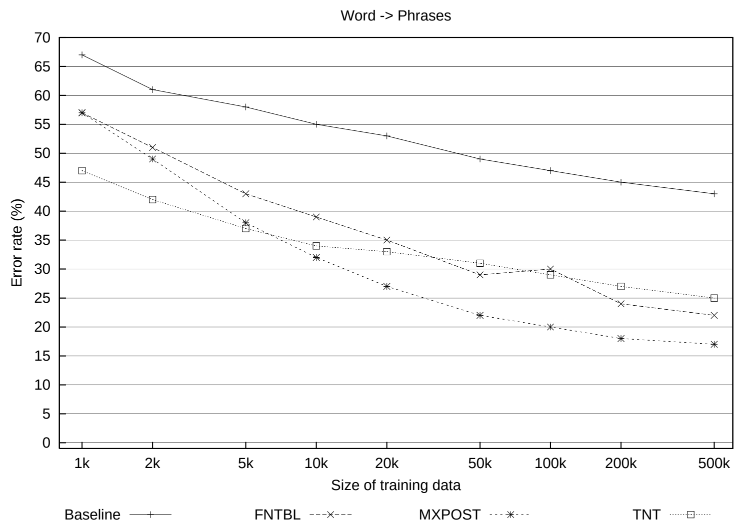

Figure 7: The error rate for each classifier when training is performed on the basis of lexical information to predict the phrase tags.

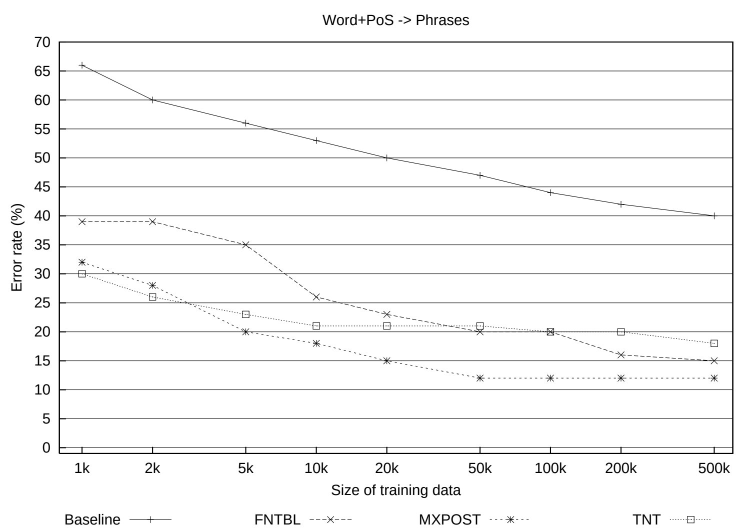

Figure 8: The error rate for each classifier when training is performed on the basis of lexical information together with the correct PoS to predict the phrase labels.

To summarize the effect of the linguistic information included in training, the best results is obtained by excluding all lexical information from the learning process. However, if the user would like to use lexical information, each token should be annotated with its PoS tag during learning, thereby eliminating homonyms, and achieving higher system performance. It is worth noting though that when using words and PoS as input, the taggers used see the

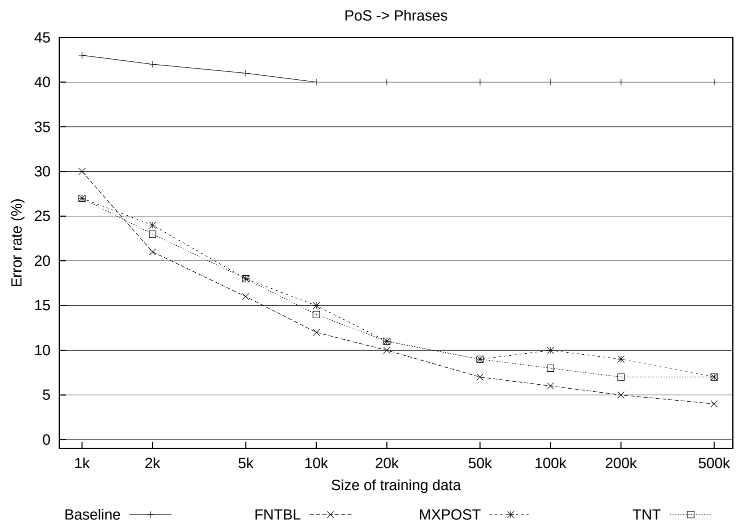

Figure 9: The error rate for each classifier when training is performed entirely on the basis of the PoS to predict the phrase labels.

input as an atom and cannot ignore the word itself. If the input features were separated from each other, the results would be different. Considering the results given when training is performed on lexical information alone, all systems perform better when recognizing the phrasal structure without the prediction of PoS tags. The statistical approaches can better learn from many different token types, while the transformation-based learner achieves highest accuracy in cases where a small number of token types is involved in learning.

Next, we will look at the effect that the size of the training set has on the three systems given the four learning tasks.

5.3 The Effect of the Size of the Training Set

As we can see from Figures 6, 7, 8, and 9, accuracy is improved for all systems by increasing the size of the training corpus. The fact that the learning task becomes easier with a larger training corpus is not surprising, since as we increase the size of the training set, we increase the number of different contextual environments in which the token types (i.e. the PoS tag, the word, or both together) can appear, as well as decrease the percentage of unknown tokens, as was shown in Figure 4. It has to be pointed out, that an additional large lexicon listing all possible classes for a token type can be used in FNTBL and TNT during learning in order to decrease the total number of unknown tokens and thereby increase system performance. However, such a lexicon was not used in this study.

The systems show different sensitivity to the size of the training corpus in the various experiments. The maximum entropy approach, MXPOST, achieves lowest error rate when large training corpus containing lexical information is used. The hidden Markov model, TNT, on the other hand, obtains fewest errors when trained on small data sets with lexical information included in training, and shows the lowest sensitivity to the size of the training set compared to the other approaches. The transformation-based approach, fntbl, succeeds well in cases when large training sets are used, especially when PoS is included as a feature type.

The reason for the different sensitivity the systems show can possibly be explained by the type of background knowledge that is implemented in the systems. The success of tnt when training is performed on small data sets might depend on the smoothing strategy incorporated for handling sparse data (in this study, linear interpolation is used when training tnt), while the other systems do not use smoothing. The success of tnt might also depend on the parameters implemented in the system for the annotation of unknown words (tnt checks up to the last ten characters of a token while the other approaches use affix analysis up to four characters only). Both the number of token types appearing only once and the number of unknown tokens are high when using small training set. For example, 77% and 86% of token types appear only once when the training data consists of 20k and 1k tokens respectively; 51% and 20% of the words are unknown in the test data given training sets including 1k and 50k tokens each (as shown in Figure 4).

The success of the maximum entropy approach, mxpost, achieving lowest error rates when the training corpus is large and includes lexical information, can be explained by the window size the system uses for disambiguation. mxpost looks at a larger window size, a context of two preceding tags, and two preceding and following tokens, while tnt uses a context of three tags, only.

The transformation-based learner, fntbl, on the other hand, does not perform well on small training sets and obtains the highest error rate in the annotation of unknown words. When only PoS categories are used as the basis for learning, we eliminate the problem of the analysis of unknown words, thereby making the classification easier for fntbl. However, this might not be the only reason for fntbl's success. The contextual environment that fntbl uses for disambiguation to predict the phrasal structure of a particular PoS tag is largest among the PoS taggers. fntbl uses a window size of up to seven tokens/tags, that is a context of up to three preceding and following tokens and/or tags.

Thus, as we have seen, the type of linguistic information used in learning, and the size of training set are both important facts that we have to consider when building data-driven chunkers/parsers. However, the reader should keep in mind that the results presented above do not show differences between the algorithms *per se*, since the algorithms are trained with different parameters. Rather, the results only let us compare the implementations of the algorithms, i.e. the PoS taggers, that are used—as they are—with their default settings for the parsing task.

5.4 The Effect of Background Knowledge

In the previous sections, clear differences were found between the parsers' performance for the various learning tasks. We hypothesized that the background knowledge the parsers use for the identification of unknown words and their disambiguation strategies may play an important role in the systems' performance. The question is whether the results obtained can be related to the properties of the algorithms *per se*, or to the parameters (such as suffix analysis or window size) used by the taggers.

In this section, we present a pilot investigation on how the parameters used in the implementation of algorithms might influence system performance. First, fntbl is trained with the same parameter settings as mxpost and tnt use (see Section 3.3). Training is carried out separately with regard to lexical parameters for the analysis of unknown tokens, contextual parameters for the disambiguation of known tokens, and the combination of both types, according to the taggers' (mxpost and tnt) settings. Other types of features, for example smoothing, was not implemented in fntbl in order to simulate tnt. Second, tnt is trained with different smoothing methods and suffix analysis. For fntbl, the change of parameters is straightforward and easily applicable while for mxpost and tnt, the source code is not included in the releases. Therefore, re-implementations of these algorithms would be necessary in order to be able to include the same parameters in each system.

In this experiment, training is performed on a small, medium, and large data set—the same as was used in the previous experiments—consisting of 2k, 20k, and 200k tokens respectively on the four types of training sets with different types of linguistic information included in learning. The test set is the same as in the experiments described above.

The results for fntbl using the parameter settings of mxpost and tnt are shown in Table 4 and Table 5 respectively when the four types of data sets of various sizes served as training data. In the second and the last columns, the parsers' original performance is shown (reported in Section 5.3) while in columns three, four and five, the results given by the simulation experiment are listed. Accuracy is also shown for known and unknown words separately in the Appendix in Table 9.

Considering the results, the original implementations of the systems have highest overall performance in all training experiments. When fntbl is trained with the same lexical and contextual parameters as mxpost (see column 3 in Table 4) with words included in learning, accuracy drops markedly compared to mxpost's original performance. On the other hand, when training is performed on PoS sequences only, fntbl's accuracy somewhat increases compared to mxpost's original results. The results indicate differences between the algorithm bias; fntbl has the advantage of learning on the basis of a few token types while mxpost learns best when a large number of types is included in the training data. However, fntbl's original performance and the results when it is trained with mxpost's lexical and its own contextual parameters (see column 4) are directly comparable. This is due to the fact that the same parameters are used by both fntbl and mxpost for affix analysis. However, when only mxpost's contextual parameters are used (see column 5), thereby decreasing the window size of fntbl from seven to five, accuracy decreases for large training sets because of the higher error rate of the annotation of known tokens (see Table 9) while performance increases somewhat for small data sets.

The accuracy rates for fntbl trained by using tnt's lexical and/or contextual parameters are shown in column 3, 4 and 5 in Table 5. Here, as was the case in the previous example, the original implementation of tnt achieves higher performance than fntbl using tnt's lexical and contextual parameter settings (see column 2 and 3). The deviance is largest for small training sets, and the error rate on average is highest for the annotation of unknown words (see Table 9) indicating that tnt has a better method for analyzing unknown tokens. By extending fntbl's lexical templates from four to up to ten characters while keeping the same window size (see column 4), performance of fntbl is directly comparable to fntbl's original results. However, when we decrease the window size instead

Accuracy (%)	MXPOST ORIGINAL			MXPOST-LEX MXPOST-CON			MXPOST-LEX FNTBL-CON			FNTBL-LEX MXPOST-CON			FNTBL ORIGINAL		
Information	2k	20k	200k	2k	20k	200k	2k	20k	200k	2k	20k	200k	2k	20k	200k
Word → *PoS* + *Ph*	40.7	62.9	77.9	43.2	58.1	70.9	43.0	59.0	72.7	42.5	58.6	70.7	44.3	58.5	72.8
Word → *Ph*	50.6	72.5	81.7	47.8	64.0	73.1	47.9	64.5	72.2	48.9	63.4	73.7	48.5	64.4	75.1
Word+PoS → *Ph*	71.3	84.6	87.9	60.0	74.7	80.9	60.5	75.5	83.2	63.4	74.5	81.8	60.8	76.2	83.3
PoS → *Ph*	75.5	88.4	90.0	76.6	87.4	91.7	78.3	89.4	94.8	76.6	87.4	91.7	78.6	89.4	94.8

Table 4: Accuracy (%) is shown for FNTBL when trained with the same parameters—either lexical (column 4), contextual (column 5), or both (column 3)—as MXPOST.

from seven to three tags (see column 5), performance decreases as the size of the training set increases due to the high error rate for the annotation of known tokens, as is shown in Table 9.

Accuracy (%)	TNT ORIGINAL			TNT-LEX TNT-CON			TNT-LEX FNTBL-CON			FNTBL-LEX TNT-CON			FNTBL ORIGINAL		
Information	2k	20k	200k	2k	20k	200k	2k	20k	200k	2k	20k	200k	2k	20k	200k
Word → *PoS* + *Ph*	49.5	63.2	72.2	43.5	56.1	64.7	44.5	58.9	72.4	43.1	56.4	64.5	44.3	58.5	72.8
Word → *Ph*	57.9	66.9	72.8	48.5	61.8	71.4	48.1	63.9	75.1	48.3	60.9	71.4	48.5	64.4	75.1
Word+PoS → *Ph*	73.1	78.5	79.9	62.5	73.0	77.0	60.8	76.2	81.3	61.9	72.8	78.1	60.8	76.2	83.3
PoS → *Ph*	76.6	88.1	92.0	73.5	81.1	83.4	78.2	89.4	94.8	73.6	81.1	83.4	78.6	89.4	94.8

Table 5: Accuracy (%) is shown for FNTBL when trained with the same parameters—either lexical (column 4), contextual (column 5), or both (column 3)—as TNT.

Lastly, previously we hypothesized that the reason for TNT's success on small corpora might be the smoothing strategy for sparse data and/or the suffix analysis of up to ten characters for unknown words. In this experiment, TNT is trained without any smoothing used in learning, as well as a lexical parameter setting using four character analysis only. The results are summarized in Table 6. It is obvious that when training is performed without any smoothing, accuracy decreases to a great extent. Without smoothing involved in learning, the other approaches would outperform TNT in the experiments described in previous sections. We can also notice that, in contrast to what was expected, a suffix analysis of up to four characters gives higher performance on average than that of ten letters.

The results indicate that both the algorithm bias and the parameter settings used in learning play an important role in system performance. However, further investigation is necessary to find out the relationship between algorithm and information bias, as was pointed out by De Pauw and Daelemans (2000). Next, we will describe how to use the results, reported in this paper, in real-world applications in an efficient way.

Accuracy (%)	TNT ORIGINAL			NO SMOOTHING 10 LETTERS			NO SMOOTHING 4 LETTERS			SMOOTHING 4 LETTERS		
Information	2k	20k	200k	2k	20k	200k	2k	20k	200k	2k	20k	200k
Word → $PoS + Ph$	49.5	63.2	72.2	30.1	43.3	52.1	35.1	43.2	52.4	49.9	65.0	73.7
Word → Ph	57.9	66.9	72.8	49.6	58.4	64.6	50.0	59.5	65.5	58.7	68.9	74.3
Word+PoS → Ph	73.1	78.5	79.9	60.5	68.3	71.5	61.9	70.0	70.6	76.7	78.9	79.8
PoS → Ph	76.6	88.1	92.0	64.1	83.9	91.6	64.1	83.9	91.6	76.6	88.2	92.1

Table 6: Accuracy (%) is shown for TNT when trained on 2k, 20k, and 200k tokens using suffix analysis of 4 and 10 characters with and without linear interpolation as smoothing strategy.

5.5 Evaluation for Real-World Applications

The reader might ask how we can apply the results described in this paper in real-world applications in which the system needs both PoS tagged and parsed text. An obvious solution is to let the best PoS tagger available for the particular language or domain annotate the text to be analyzed. The next step would be to extract the PoS labels from the text but keep the sentence division, and let the parser annotate the PoS sequences. The only thing then remaining to do would be to put the words back into the parsed PoS sequences.

Obviously, if the user does not have the text annotated with correct PoS tags but has to use a tagger for that purpose, the performance of the parser can be expected to decrease. Therefore, an evaluation for real-world applications appears to be necessary.

Since the results described above show that the most successful parsing classification is achieved by training on PoS categories only to predict the constituent structure of a token, the parsers which were trained entirely on PoS information were used for the chosen real-world evaluation task. First, the unannotated test data was tagged by a PoS tagger. For that purpose, TNT is used since this tagger has been proved to achieve highest accuracy on larger training sets for Swedish (Megyesi, 2001). Second, the words were removed from the PoS tagged text, and the PoS sequences were labeled with phrase categories by each parser. The PoS tagger TNT, as well as the three parsers (given when only PoS sequences were included in training) were trained on 500k tokens. The reason for training the PoS tagger, TNT, on 500k correctly annotated tokens only (and not on the whole SUC corpus) was to assure not to include any of the test sentences into the training data.

The results are shown in Table 7. The performance of the PoS tagger is 94.98%. As was expected, the performance of the parsers is considerably lower than was reported for correct PoS sequences in Section 5.3. FNTBL achieves the highest accuracy, followed by MXPOST and TNT, just as in the fourth experiment, described in Sections 5.1 and 5.2.

Thus, we have seen that the morphological and shallow syntactic annotation of words in texts, including PoS with morphological features and the whole hierarchical phrase structure that the word belongs to, is possible with approximately 90% correct result.

TAGGER	PARSER	RESULT
94.98%	FNTBL	**90.83%**
	MXPOST	88.87%
	TNT	88.19%

Table 7: Accuracy (%) is given for each classifier when the test set was first tagged by the PoS tagger TNT, then parsed with the three classifiers based on PoS sequences.

6. Conclusion and Future Work

This paper has presented empirical results on the application of publicly available PoS taggers to shallow parsing of Swedish texts. The PoS taggers included in the study are the transformation-based learner FNTBL, the maximum entropy approach MXPOST, and the hidden Markov model Trigrams'n'Tags (TNT). The goal of the shallow parsers is to recognize the constituent structure of the sentence, representing the whole hierarchical structure the token belongs to in the parse tree. The encoding is based on the concatenation of the phrase tags on the path from lowest to higher nodes. The results show that the data-driven, language and tag set independent PoS taggers can be efficiently used for shallow parsing of texts, given that PoS information only, i.e. without the presence of words, is included in the training data.

Several aspects of the classifiers were evaluated, such as the taggers' sensitivity to certain kinds of linguistic information included in the training data. Particular attention has been directed to the various types of input features that the taggers learn from, such as words, PoS tags, and a combination of both. Also, experiments have been carried out on various numbers of target classes that the taggers have to search through in order to predict phrasal categories only, or to recognize both PoS and the phrasal structure of the tokens. Furthermore, the taggers' sensitivity to the size of the training set including different linguistic information was investigated. Additionally, some of the effects of the parameter settings used by PoS taggers have been examined.

The results show that for all three systems, best performance is obtained if the number of token types the taggers learn from is reduced by only considering the PoS tags. By excluding the lexical information during learning and testing, all classifiers obtain an accuracy above 92% when the training set contains at least 50k tokens.

However, the type of linguistic information, the size of the training set, the algorithms, as well as the parameters used by the algorithms are all factors that influence system performance. This study shows clear differences between the taggers' sensitivity to the type of information used in learning, and the number of target classes to be learned.

The transformation-based learner, FNTBL, obtains best results when training is performed on PoS categories alone. However, FNTBL does not succeed as well as the statistical approaches in the analysis of the unknown tokens.

The maximum entropy learner, MXPOST, is most successful on average when training is performed on large data sets containing lexical information with or without PoS tags.

mxpost succeeds best among the three systems when the training data includes many different token types.

The hidden Markov model based tnt outperforms all systems when the size of the training set is small and includes lexical information with or without PoS information included in training. When small training set is used, the percentage of unknown tokens is considerably large, making classification more difficult.

The taggers described in this study were used with their default settings, including different types of background knowledge for the analysis of unknown words and the disambiguation of known words. Also, a pilot study was carried out training fntbl with the same lexical and/or contextual parameters as mxpost and tnt use. Furthermore, tnt was trained with the same lexical parameters as fntbl and mxpost use with and without smoothing. The preliminary results show that both the parameters implemented in the systems, the algorithm bias and the smoothing involved in learning play an important role in the systems' performance.

Future work includes a careful investigation of the algorithms applied to parsing when using the same parameter settings (context window size and number of characters) for each algorithm trained on various linguistic information. Additionally, since the algorithms were not optimized for Swedish, it would be necessary to investigate the best combination of features in the parameter settings of the systems.

Future work also includes the improvement of the detection of maximal projections, and the provision of automatic data-driven detection of clause boundaries, such as relative clauses and other subordinate clauses for Swedish. In order to further improve parsing accuracy, the best combinations of the approaches could be determined by constructing ensembles of classifiers.

Lastly, since the method described in this paper can be assumed to be directly applicable to other languages as well, it would be very interesting to find out how well it would perform on various language types including languages with complex morphology and free word order.

Acknowledgments

I am very grateful to my supervisor Rolf Carlson, the three anonymous reviewers for their valuable and helpful comments on the draft manuscript, and all the researchers who created the PoS taggers used in this study including Thorsten Brants, Radu Florian, Grace Ngai, and Adwait Ratnaparkhi. Also, I would like to thank Anette Hulth, Sara Rydin, Mattias Heldner, and Fredrik Olsson for discussions and comments, and Sheri Hunnicutt for proof-reading. Whatever errors remain are, of course, all mine. This research was supported by vinnova, ctt's industrial partners, kth, and hp Voice Webb Initiative.

Appendix

Feature	Class	Rate	ADVP	AP	APMAX	INFP	NP	NPMAX	NUMP	PP	VC
	Total		5,970	10,477	1,433	2,541	53,810	24,350	1,951	29,419	16,282
W → $PoS + Ph$	FNTBL	P	84.16	77.01	18.74	86.24	95.20	78.29	95.84	86.00	93.31
		R	82.91	71.14	11.65	88.55	93.12	76.39	85.08	84.03	93.59
		F-B1	83.53	73.96	14.37	87.38	94.15	77.33	90.14	85.00	93.45
	MXPOST	P	81.26	84.97	50.21	94.34	96.83	87.90	92.22	92.87	94.36
		R	81.16	72.55	16.47	93.23	94.49	72.34	86.93	84.36	97.90
		F-B1	81.21	78.27	24.80	93.78	95.65	79.36	89.50	88.41	96.10
	TnT	P	83.41	82.98	41.72	88.09	97.04	73.64	93.95	82.95	95.97
		R	83.52	77.78	20.03	89.65	95.61	61.18	89.90	75.91	97.68
		F-B1	83.46	80.29	27.07	88.86	96.32	66.83	91.88	79.27	96.82
W → Ph	FNTBL	P	82.95	74.43	34.34	87.32	95.09	79.55	95.97	87.53	92.04
		R	82.95	74.26	16.68	85.40	93.32	69.89	86.57	78.95	95.04
		F-B1	82.95	74.34	22.45	86.35	94.20	74.41	91.03	83.02	93.52
	MXPOST	P	84.41	86.17	61.22	96.61	95.88	88.78	94.44	93.88	94.16
		R	77.74	73.17	18.84	84.14	95.19	72.84	85.39	84.10	97.27
		F-B1	80.93	79.14	28.81	89.94	95.53	80.02	89.69	88.72	95.69
	TnT	P	82.02	80.78	36.53	87.82	95.98	73.88	93.53	83.15	95.01
		R	81.12	77.75	22.61	86.86	95.04	60.15	89.60	73.96	96.90
		F-B1	81.57	79.24	27.93	87.34	95.51	66.32	91.52	78.29	95.95
W+PoS → Ph	FNTBL	P	99.70	85.15	48.45	94.82	97.77	84.98	96.84	90.79	99.12
		R	98.84	86.93	35.80	92.99	96.99	77.40	94.31	85.19	99.31
		F-B1	99.27	86.03	41.18	93.90	97.38	81.01	95.56	87.90	99.21
	MXPOST	P	98.79	93.08	79.02	98.78	98.95	93.45	96.56	95.93	99.15
		R	98.14	85.45	33.64	95.79	97.57	77.25	92.11	86.43	99.74
		F-B1	98.46	89.10	47.19	97.26	98.26	84.58	94.28	90.93	99.44
	TnT	P	99.23	90.59	56.62	91.61	98.45	79.44	98.25	87.14	99.30
		R	99.65	88.01	34.61	91.97	97.45	67.67	95.23	79.24	98.99
		F-B1	99.44	89.28	42.96	91.79	97.95	73.08	96.72	83.00	99.14
PoS → Ph	FNTBL	P	78.75	95.18	96.09	100.00	99.39	97.84	98.65	98.59	100.00
		R	82.36	95.49	78.86	99.96	99.12	97.34	97.44	97.62	99.99
		F-B1	80.51	95.33	86.63	99.98	99.25	97.59	98.04	98.10	99.99
	MXPOST	P	78.13	95.44	85.28	99.84	99.34	96.70	97.41	96.90	99.89
		R	77.19	88.57	67.13	98.07	98.12	79.47	96.41	89.89	99.99
		F-B1	77.66	91.88	75.12	98.95	98.73	87.24	96.91	93.26	99.94
	TnT	P	67.75	94.06	79.49	96.87	99.12	95.55	98.71	96.78	100.00
		R	88.58	93.79	76.55	99.76	98.63	95.89	98.05	96.55	99.68
		F-B1	76.78	93.92	77.99	98.29	98.87	95.72	98.38	96.66	99.84

Table 8: Precision (P), recall (R) and $F_{\beta=1}$ (F-B1) rates for each classifier when trained on 200k tokens on the four types of input features, and tested on 117,530 tokens.

FEATURE	PARAMETER		2K			20K			200K		
	LEXICAL	CONTEXT	T	K	U	T	K	U	T	K	U
W → $PoS + Ph$	FNTBL	original	44.3	67.3	15.6	58.5	70.8	25.8	72.8	77.6	36.9
	MXPOST	original	40.7	56.2	21.3	62.9	70.5	42.5	77.9	80.1	61.2
	TnT	original	49.5	68.9	25.2	63.2	71.1	42.1	72.2	75.0	51.9
	FNTBL	MXPOST	42.5	66.8	12.1	58.6	69.9	28.4	70.7	75.1	38.3
	MXPOST	FNTBL	43.0	67.1	12.9	59.0	71.0	27.0	72.7	77.6	36.2
	MXPOST	MXPOST	43.2	66.6	14.0	58.1	70.0	26.1	70.9	75.3	38.1
	FNTBL	TnT	43.1	65.8	14.8	56.4	67.4	27.1	64.5	71.2	14.6
	TnT	FNTBL	44.5	67.8	15.4	58.9	71.1	26.1	72.4	77.5	34.4
	TnT	TnT	43.5	65.7	15.7	56.1	67.3	26.2	64.7	71.6	12.9
W → Ph	FNTBL	original	48.5	70.4	21.0	64.4	79.6	45.5	75.1	77.9	54.4
	MXPOST	original	50.6	63.9	33.9	72.5	77.2	59.8	81.7	83.0	72.0
	TnT	original	57.9	72.8	39.4	66.9	72.7	51.4	72.8	75.0	56.3
	FNTBL	MXPOST	48.9	70.6	24.0	63.4	73.3	36.9	73.7	78.1	41.3
	MXPOST	FNTBL	47.9	70.6	19.7	64.5	73.4	40.7	72.2	77.5	33.0
	MXPOST	MXPOST	47.8	57.8	20.8	64.0	73.4	38.9	73.1	77.9	37.3
	FNTBL	TnT	48.3	70.2	21.0	60.9	71.2	33.4	71.4	74.7	47.0
	TnT	FNTBL	48.1	70.8	19.8	63.9	73.3	38.8	75.1	77.9	54.4
	TnT	TnT	48.5	70.4	21.2	61.8	71.9	34.8	71.4	74.7	47.1
W+PoS → Ph	FNTBL	original	60.8	74.8	44.8	76.2	79.3	68.5	83.3	83.5	82.4
	MXPOST	original	71.3	79.0	62.6	84.6	86.7	79.3	87.9	88.4	84.3
	TnT	original	73.1	79.7	65.5	78.5	78.9	77.5	79.9	80.0	79.5
	FNTBL	MXPOST	63.4	77.5	47.3	74.5	79.1	63.1	81.8	82.6	76.7
	MXPOST	FNTBL	60.5	74.8	44.2	75.5	79.1	66.3	83.2	83.6	80.0
	MXPOST	MXPOST	60.0	77.6	39.8	74.7	79.1	63.8	80.9	82.6	69.8
	FNTBL	TnT	61.9	76.9	44.7	72.8	76.7	63.1	78.1	79.7	67.1
	TnT	FNTBL	60.8	74.8	44.5	76.2	79.4	68.1	81.3	83.4	66.8
	TnT	TnT	62.5	77.3	45.7	73.0	76.8	63.4	77.0	79.6	59.0
PoS → Ph	FNTBL	original	78.6	78.9	49.7	89.4	89.5	15.5	94.8	94.8	70.4
	MXPOST	original	75.5	75.8	47.1	88.4	88.5	38.0	90.0	90.0	20.0
	TnT	original	76.6	76.8	50.2	88.1	88.2	33.8	92.0	92.1	40.0
	FNTBL	MXPOST	76.6	76.9	47.3	87.4	87.5	22.5	91.7	91.7	00.0
	MXPOST	FNTBL	78.3	79.0	8.6	89.4	89.5	19.7	94.8	94.8	00.0
	MXPOST	MXPOST	76.6	76.9	47.3	87.4	87.5	0.0	91.7	91.7	00.0
	FNTBL	TnT	73.6	73.9	44.1	81.1	81.1	18.3	83.4	83.4	00.0
	TnT	FNTBL	78.2	79.0	0.0	89.4	89.5	19.7	94.8	94.8	00.0
	TnT	TnT	73.5	73.8	37.4	81.1	81.1	0.0	83.4	83.4	00.0

Table 9: The results are given for FNTBL using different lexical and contextual parameters of MXPOST and TNT, trained on 2k, 20k and 200k tokens on the four types of input features, and tested on 117,530 tokens. Accuracy (%) is calculated for the total number of tokens (T), as well as for known (K) and unknown (U) tokens.

References

Steven Abney. Parsing by Chunks. In *Principle-Based Parsing*. Kluwer Academic Publishers, 1991.

Shlomo Argamon, Ido Dagan and Yuval Krymolowsky. A Memory-Based Approach to Learning Shallow Natural Language Patterns. In *Proceedings of 36th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 67-73, Montreal, Canada, 1998.

John Aycock. Compiling Little Languages in Python. In *Proceedings of the 7th International Python Conference*, 1998.

Thorsten Brants. Cascaded Markov Models. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL-99)*, Bergen, Norway, 1999.

Thorsten Brants. TnT - A Statistical Part-of-Speech Tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference*, Seattle, Washington, USA, 2000.

Eric Brill. Automatic Grammar Induction and Parsing Free Text: a Transformation-Based Approach. In *Meeting of the Association for Computational Linguistics (ACL)*, pp. 259-265, 1993.

Eric Brill. Some Advances in Rule-Based Part of Speech Tagging. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, Seattle, Washington, 1994.

Sabine Buchholz, Jorn Veenstra and Walter Daelemans. Cascaded Grammatical Relation Assignment. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

Claire Cardie and David Pierce. Error-Driven Pruning of Treebank Grammars for Base Noun Phrase Identification. In *Proceedings of COLING/ACL*, pp 218-224, Montreal, Canada, 1998.

Kenneth Church. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Texts. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pp. 136-143. Association for Computational Linguistics, 1988.

James Cussens. Notes on Inductive Logic Programming Methods in Natural Language Processing (European Work). Manuscript (http://www.cs.york.ac.uk/mlg/), 1998.

Walter Daelemans, Jakub Zavrel, Peter Berck and Steven E. Gillis. MBT: a Memory-Based Part of Speech Tagger-Generator. In *Proceedings of Fourth Workshop on Very Large Corpora (VLC-96)*, pp. 14-27, Copenhagen, Denmark, 1996.

Walter Daelemans, Antal van den Bosch and Jakub Zavrel. Forgetting exceptions is harmful in language learning. In *Machine Learning*, 34, 1999.

Guy De Pauw and Walter Daelemans. The Role of Algorithm Bias vs Information Source in Learning Algorithms for Morphosyntactic Disambiguation. In *Proceedings of Computational Natural Language Learning (CoNLL-00)*, pp. 19-24, Lisbon, Portugal, 2000.

Martin Eineborg and Nikolaj Lindberg. ILP in part-of-speech tagging - An overview. In Cussens, J and Dzeroski, S (editors), *Learning Language in Logic Workshop (LLL99)*, Bled, Slovenia. 2000.

Eva Ejerhed, Gunnel Källgren, Ola Wennstedt and Magnus Åström. *The Linguistic Annotation System of the Stockholm-Umeå Project.* Dept. of General Linguistics, University of Umeå, 1992.

James Paul Gee and François Grosjean. Performance Structures: A psycholinguistic and linguistic appraisal. In *Cognitive Psychology*, 15, pp. 411-458, 1983.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz and Britta Schasberger. The Penn Treebank: A Revised Corpus Design for Extracting Predicate Argument Structure. In *Human Language Technology, ARPA March 1994 Workshop*, Morgan Kaufmann, 1994.

Beáta Megyesi. Comparing Data-Driven Learning Algorithms for PoS Tagging of Swedish. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2001).* pp. 151-158, Carnegie Mellon University, Pittsburgh, PA, USA, June, 2001.

Beáta Megyesi. Phrasal Parsing by Using Data-Driven PoS Taggers. In *Proceedings of Recent Advances in Natural Language Processing (EuroConference RANLP-2001)*, Tzigov Chark, Bulgaria, September, 2001.

Grace Ngai and Radu Florian. Transformation-Based Learning in the Fast Lane. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL-2001)*, pp. 40–47, June, 2001.

Miles Osborne. Shallow Parsing as Part-of-Speech Tagging. In *Proceedings of CoNLL-2000 and LLL-2000*, pp. 145-147, Lisbon, Portugal, 2000.

Lance A. Ramshaw and Mitchell P. Marcus. Text Chunking Using Transformation-Based Learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, Association for Computational Linguistics, 1995.

Adwait Ratnaparkhi. A Maximum Entropy Model for Part-of-Speech Tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-96)*, Philadelphia, PA, USA, 1996.

Wojciech Skut and Thorsten Brants. Chunk Tagger Statistical Recognition of Noun Phrases. In *ESSLLI-98 Workshop on Automated Acquisition of Syntax and Parsing (ESSLLI-98)*, Saarbrücken, Germany, 1998.

Erik Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of CoNLL and LLL-2000*, pp. 127-132, Lisbon, Portugal, 2000.

Hans van Halteren (editor). *Syntactic Wordclass Tagging*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

Jorn Veenstra. Memory-Based Text Chunking. In *Workshop on Machine Learning in Human Language Technology*, ACAI-99, Crete, Greece, 1999.

Jakub Zavrel and Walter Daelemans. Recent Advances in Memory-Based Part-of-Speech Tagging. In *Proceedings of the VI Simposio Internacional de Comunicacion Social*, pp. 590-597, Santiago de Cuba, 1999.