Shallow Parsing with PoS Taggers and Linguistic Knowledge
A Comparative Study of Three Algorithms and Four Learning Tasks

Beáta Megyesi
Centre for Speech Technology
Department of Speech, Music and Hearing
KTH, Sweden
Drottning Kristinas väg 31
SE-100 44, Stockholm, Sweden

Editor:

Abstract

In this study, three data-driven algorithms are applied to shallow parsing of Swedish texts by using state-of-the-art data-driven PoS taggers as the basis for parsing. 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 a hierarchical fashion. A special attention is directed to the algorithms’ sensitivity to different types of linguistic information included in the training data, as well as the algorithms’ sensitivity to the size of the various types of training data sets. Four types of linguistic features are used; the algorithms are trained on the basis of lexical information only, part-of-speech only, and both, to predict the phrase structure of the tokens with or without part-of-speech. The results show that best performance can be obtained by training on the basis of PoS tags with labels marking the phrasal constituents without considering the words themselves.

1. Introduction

Machine learning techniques in the last decade has permeated through several areas of natural language processing (NLP). The reason is that a vast number of machine learning algorithms have been proved to be able to learn from natural languages data given a relatively small amount of correctly annotated data set. Therefore, machine learning algorithms make 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 tag-
gers is that they are language and tag set independent and thereby are easily applicable to new languages and domains. The average accuracy that are 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, the disambiguation of words according to their context. The disambiguation is already taken care of by the PoS taggers which use some kind of background knowledge; parameters that tell the system to check the contextual environment of the current word and or tag.

As a first step to build corpus-based parsers, a considerable amount of research has been performed 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 as 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 data-driven chunking, a lot of attention has been directed towards the development of the recognition of simple, non-recursive noun phrases, also called base NP chunks (e.g. Cardie and Pierce (1998); Church (1988); Skut and Brants (1998)). These phrases play an important role in many application areas, like information extraction and information retrieval, as well as in human language processing (Gee and Grosjean, 1983). Research on the detection of other chunk types, like prepositional phrases (PP), adverb phrases (ADVP), adjective phrases (ADJP) and verb clusters, by data-driven methods have been also performed with promising results (Argamon et al. (1998); Brants (1999); Buchholz et al. (1999); Megyesi (2001a); Osborne (2000); Ramshaw and Marcus (1995); Veenstra (1999)). 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, almost in all these studies with the exception of a 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 is 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 to mention that the majority of studies on chunking has been focused on the development of data-driven chunkers/parsers for English, just as it was in the case of 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 corpus is missing for most of the languages. Given this 'correctly' parsed large data set, the development and evaluation of the data-driven approaches become easier and reliable.
The goal of this work, reported in this paper, is to build data-driven chunker/shallow parser for Swedish without a lot of human effort. Desirable properties of the shallow parser are as follows:

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

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 a correctly chunked/parsed 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 basis to parse Swedish texts. The algorithms are 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 and what effects do different kind of linguistic information included in the training data have on the different machine learning algorithms in this processing. A pilot study for this task was performed and reported by Megyesi (2001a).

In this study, shallow parsers are developed for Swedish by using data-driven PoS taggers and linguistically well-motivated labels describing the whole constituent structure the word belongs to in a hierarchical structure. We will describe the algorithms' sensitivity to different type of linguistic information included in the training data and features to be used in training so that the algorithms efficiently can learn to parse texts.

The remainder of the paper is organized as follows: Section 2 gives an overview on previous studies performed on data-driven chunking; Section 3 presents the data-driven shallow parsers where the linguistic categories used to shallow parse texts, the training and test data, and a brief description of the algorithms that the parsers are built on are presented; Section 4 describes the experiments when various linguistic features are used in learning; Section 5 presents the results; Section 6 discusses the results and gives directions to further research; Finally, section 7 concludes the paper.

2. Previous Work on Data-Driven Text Chunking

The concept of 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 the 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 it 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 baseNPs, to the nominal head, including determiners and adjectives, but not prepositional phrases or other type of arguments located after the head word. The tag set consisted of three types of tags: I for the words inside the baseNP, O for the words outside of the noun phrase, and B for the first word of the chunk. The second target of their work was to partition sentences into non-overlapping noun-type and verb-type chunks in a similar fashion as was proposed by Abney (1991). The noun-type chunks consisted of, among others, noun phrases to the nominal head, prepositional phrases including an NP argument, but not coordinating conjoined NPs. Each N and V type had two tags, depending on if 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 derive automatically 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 results showed a recall of 93.5% and a precision of 93.1% for baseNP chunks when trained on 950k words and tested on 50k words. They achieved 88% for partitioning the sentence into N and V chunks when trained on 200k words. Also, they pointed out that training set size 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%, thus lower than that of Ramshaw and Marcus.

Also other experiments on data-driven chunking were performed with memory-based learning method. 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 by using local repair heuristics that improved the precision without decreasing the recall as well as by discarding the ten worst rules without decreasing the precision. They achieved 94% precision and recall on simple baseNPs, and 91% on more complex ones.

Veenstra (1999), also using memory-based learning learning technique, tgrree (Daelemans et al., 1996), described experiments on NP, VP and PP chunking using the Wall Street Journal for data and the BIO labels proposed by Ramshaw and Marcus (1995) attached to each chunk type as it was proposed by Ramshaw and Marcus (1995) and described above. He reported precision and recall rates of 94%-95%, and accuracy of 98% and 99% for NP and VP chunks, respectively.
Buchholz et al. (1999) used the memory-based learning to assign grammatical relations (e.g. 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 divided into several steps. First prepositions, NP, VP, ADJP and ADVP chunks were found simultaneously, then prepositions and NPs were collapsed into PPs. They reported precision and recall rates of 92% for NP and VP chunks, 95% and 96% for prepositions, 68% for AP chunks, and 78% for ADVP chunks. For PP chunk, the precision and recall rates were 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 algorithms was tested on 300k words taken from the NEGRA corpus consisting of German news paper texts. Recall was between 54% and 84.8% for 1 resp. 9 layers, and 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 also 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 FB1 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). The training and test data for this task was taken from the Wall Street Journal corpus (WSJ). The training data consisted of 211,727 tokens and the test data of 47,377 tokens. The types of chunks used in the study was the same as the CoNLL-2000 shared task on chunking (Tjong Kim Sang & Buchholz, 2000). The tag set included '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 were to identify base phrase categories. Next, a description of the shallow parsers will be presented: the hierarchical phrase representation, the training data and benchmark, and a brief description of the algorithms.

3. Building Shallow Parsers

Three different aspects are needed to be addressed to be able to build a data-driven shallow parser; algorithm(s), training and test data, and classes that the algorithms have to learn to predict. In the following sections, these aspects will be described in reversed order.

3.1 Phrase Structure Representation

As we have seen in section 2, 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 is 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 dialogue systems or text-to-speech systems, where information about the whole constituent structure might 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 to 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, like in 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 of categories correspond to the chunks used in previous studies, e.g. 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 to handle co-ordinated phrases, such as maximal adjective phrases. 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, numerical expressions or verbs.
  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

- 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 like in the CoNLL-2000 shared task on chunking.
  e.g. Pilger's very interesting and nice book

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

- Maximal Projection of NP (NPMAX) includes an NP with a following PP as a modifier.
  e.g. Pilger's very interesting and nice book 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

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

Please note, that the grammatical categories do not represent neither clauses, such as relative clauses, nor sentences. This 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 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 (SVO) 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 benchmark corpus. For the treebank development, an Earley Parser, SPARK (Aycock, 1998) together with a context-free grammar for Swedish developed by the author, was used.

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 as three types of tags marking positions in a similar way as it was proposed by Ramshaw and Marcus (1995) and used in the CoNLL-2000 competition:

\[ \text{XB} - \text{the first word of the phrase } X \]

\(^1\) 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.
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 sentence 'Everybody should read Pilger's very good books about politics' represented first by parenthesis, and second by tags.

```
[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].

Alla/NPB (everybody)
borde/VCB (should)
läsä/VCI (read)
Pilgers/NPB, NPMAXB (Pilger's)
mycket/ADVPB_APB_NPMAXI (very)
bra/APB_API_NPMAXI (good)
böcker/NPB_NPMAXI (books)
om/PPB_NPMAXI (about)
politik/NPB_PLL_NPMAXI (politics)
./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 the 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 a small percentage of errors in both the training and benchmark. Unfortunately, the manual post-processing of the elimination of the noise is very time-consuming so the author decided not to correct these errors.

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

### 3.3 Algorithms

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 that they are language and tag set independent, freely available for research and have been successfully
used for several languages. The taggers that will be used to shallow 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 Hidden Markov Model (Brants, 2000). Each approach will be briefly described.

FNTBL developed by Ngai and Florian (2001), is a fast version of Brill’s transformation-based learning algorithms. 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 (e.g. the most frequently occurring tag in the corpus). Then, an ordered list of rules learned during training are 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, TBL 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 simultaneously used in order to predict the PoS tags. The binary features used by ME as default include the current word, the following and preceding two words and the preceding 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) is a statistical approach, based on a hidden Markov model and 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, i.e. up to the last ten letters 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 training data described in section 3.2.

4. Experiments on Various Linguistic Features in Learning

In previous studies on chunking, different type 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 PoS information, Argamon et al. (1998) and Cardie and Pierce (1998) learned on the basis of PoS sequences without including any lexical information, while Brants (1999) entirely utilized the words to recognize both the PoS tags and the chunks.

---

2. The difference between Brill's original implementation and Ngai & Florian's implementation of TBL is that the later stores the rules in memory instead of regenerating the rules each time of the learning process. A detailed description can be found in Ngai and Florian (2001).
Table 1: Possible combinations of the linguistic features in learning.

<table>
<thead>
<tr>
<th>TYPES TO LEARN FROM</th>
<th>CLASSES TO PREDICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Words</td>
<td>PoS - Phrases</td>
</tr>
<tr>
<td>Words</td>
<td>Phrases</td>
</tr>
<tr>
<td>Words – PoS</td>
<td>Phrases</td>
</tr>
<tr>
<td>PoS</td>
<td>Phrases</td>
</tr>
</tbody>
</table>

Comparing the results of these studies, the average accuracy is reported to be lowest when training is performed on PoS sequences only. It is, however, difficult to compare the results because either the learning algorithms, the data set or the language vary across the studies. Therefore, it is of particular interest to train, test and compare the algorithms on different types of data sets containing various linguistic features and using the same training and test set for reliable evaluation.

In order to find out how well different machine learning algorithms can learn the whole hierarchical constituent structure of the words, and to examine what effect different kind of linguistic information included in the training data have on the algorithms, four experiments are carried out. Each algorithm is trained on four types of data set, each including different types of linguistic information, as it 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, only the word and its phrase tags are included in the training corpus. 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, 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 classes (phrases with or without PoS) are examined.

In order to examine how the size of the training data influences the performance of the classifiers, each algorithm is trained in each experiment nine times on the 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, are annotated by each classifier. In each experiment, the training and test sets are disjoint.

5. Results

In this section, the results from the four experiments 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. A correct parse requires complete and correct phrase labels for a token including the position information. If the word would miss a label for a phrase that it is part of, or if a phrase label would have wrong position information (BIO tag) then the whole tag is considered to be incorrect.

\[
Accuracy = \frac{\text{No. of correctly tagged tokens}}{\text{Total no. of tokens}}
\]  

(1)
Figure 1: The number of types and classes for various training data size.

Unfortunately, the, of many researchers preferred, recall and precision measures are not possible to give in this study, since the number of classes that the algorithms have to learn are large due to the hierarchical annotation. The number of possible phrase combinations lies between 400 and 3100 classes, depending on the size of the training data. In Figure 1, the number of classes and the token types are illustrated (shown as curves) for each experiment for the nine training data sets of various sizes (shown as data points). As we can see, the number of token types increases with the size of the training data in all four experiments. However, the increase is lowest when training is performed on the basis of PoS sequences only without the presence of the words. That is not surprising since the total number of PoS tags is between 82 and 143 depending on the size of the training data. It is also worth to note 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 lexically ambiguous tokens because of the presence of the PoS tags.

Considering the number of classes that the algorithms have to learn to predict, the largest number can we find when the combination of PoS and phrase labels are required by the classifiers. The number of classes in this case lies between 264 and 3099 tags.

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 2 illustrates the percentage of unknown tokens in the test set compared to the different sizes of training data in the four various experiments. Unsurprisingly, the number of unknown tokens are very low or absent when entirely PoS tags are included in the training data. The largest number of unknown tokens can we find 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 data containing both lexical and PoS information is used, the amount of unknown words became lower compared to when training is performed on lexical information only.

Baseline performance are obtained for the test data of the four types of experiments. The baseline is counted in different ways depending on the input the learners get and the
Figure 2: The percentage of unknown tokens in the test data compared to the training data in the four experiments with different training data sizes.

Figure 3: Baseline performance for each experiment for training data of various sizes.

class they have to learn to predict. Each token in the test data receives a class (i.e. either PoS & Phrases, or Phrases) label that is most frequently associated with that token type in the training data. In Figure 3, the results are shown for the training data of various sizes within each experiment type. In average, baseline performance is lowest when lexical information is involved in training. When PoS categories are also included in the training set, higher baseline performance can be achieved. We can also notice that the size of the training data influences the accuracy; when training is performed on large training corpora, baseline accuracies for the four types of training sets become even.

With these prerequisites in mind, I will describe the results given by the classifiers for each experiment.
5.1 Performance of the Classifiers

To get an overall picture of the parsers' performance, the accuracy of each classifier when training is performed on 200k tokens, is listed in Table 2. 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 algorithms 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 to a great extent.

The best performance is achieved by transformation-based learner, fntBL when only PoS information is included in training, while in the other experiments, the maximum entropy approach MXPOST obtains the best result.

However, these numbers can not tell us how sensitive the algorithms are to the size of the training data when different type of information is used in learning. An obvious guess is that the larger amount of data we use, the better performance we get, but the improvement does not necessarily have to be the same for the algorithms when we train them on various input features.

Table 2: The accuracy of each parser trained on 200k tokens is shown with four types of training data.

<table>
<thead>
<tr>
<th>TYPES</th>
<th>number</th>
<th>CLASSES</th>
<th>number</th>
<th>fntBL (%)</th>
<th>MXPOST (%)</th>
<th>TNT (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Words</td>
<td>35611</td>
<td>PoS - Phrases</td>
<td>2492</td>
<td>67.59</td>
<td>77.86</td>
<td>72.23</td>
</tr>
<tr>
<td>Words</td>
<td>35611</td>
<td>Phrases</td>
<td>534</td>
<td>67.82</td>
<td>81.66</td>
<td>72.79</td>
</tr>
<tr>
<td>Words - PoS</td>
<td>37870</td>
<td>Phrases</td>
<td>534</td>
<td>71.33</td>
<td>87.89</td>
<td>79.93</td>
</tr>
<tr>
<td>PoS</td>
<td>141</td>
<td>Phrases</td>
<td>534</td>
<td>94.84</td>
<td>90.03</td>
<td>92.07</td>
</tr>
</tbody>
</table>

5.2 What to Include in the Training Data?

The results for each experiment on the algorithms' sensitivity to the input feature sets (word, word and PoS, PoS only), and to the number of classes the algorithms learn to recognize (phrases with or without PoS), are shown in Figure 4, 5, 6 and 7, respectively.

All systems in all four experiments outperform the baseline when the training data is larger than 2k tokens. In cases, when the training data is of small size and the test set consists of a large number of unknown tokens, accuracy is as high as the baseline but never lower.

The first experiment, 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 hardest classification task for every algorithm, see Figure 4. This is not surprising since the algorithms have to learn a great number of classes, between 264 and 3099 tags, depending on the size of the training data. 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 on small training data while MXPOST outperforms TNT when using above 50k tokens for training. However, fntBL achieves higher performance than MXPOST on small data sets. The success of TNT in the PoS and phrase
label assignment when training is performed on small data set depends on the parameters used for the annotation of unknown words. The number of the unknown tokens when using small training data is high (51% for 1k tokens, and 20% for 50k tokens, respectively, see Figure 2), why good morphological analysis is needed. For the analysis of unknown words, TNT checks up to the last ten characters of a token while the other approaches use suffix analysis up to four characters only.

![Graph](image)

**Figure 4:** The error rate for each classifier when training is performed on the basis of lexical information to classify PoS and phrase structure information.

In the second experiment where PoS information is not present in the training data, i.e. the training is performed entirely on lexical information (WORD -> PHRASES), the hypothesis 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 5. Similarly to the first experiment, the maximum entropy approach, MXPOST, have lowest error rate in cases where the training corpus consists of at least 5k tokens. TNT achieves the best result on small data sets, while fnTBL have highest error rate.

In the third experiment, where both lexical and PoS information is included as input feature for the recognition of the phrasal structures (WORD & PoS -> PHRASES), the average performance of the classifiers further increases, see Figure 6. A possible explanation to the increase in systems’ performance can be that, although in this experiment we can find the largest amount of token types, the problem of lexical homonymy does not exist, since every token type becomes unique with 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 on large training data, and TNT succeeds well on small data sets. fnTBL does not succeed as well as the other two algorithms.

Lastly, in the fourth experiment, where the lexical information is not present in training (PoS -> PHRASES), the performance of the systems increases greatly, compared to cases where lexical information is included in the training process. This can be explained by the fact that the number of unknown tokens, i.e. POS tags, is very low in the case of small
Figure 5: The error rate for each classifier when training is on the basis of lexical information to predict the phrase tags.

Figure 6: 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.

training data, and non-existing when large training data size is used, see Figure 2. The baseline performance therefore increases and the learning curves of the algorithms converge, see Figure 7. fntBL obtains best accuracy, compared to the statistical approaches, TNT and MXPOST. We can find an explanation to TBL success in the background knowledge, the algorithm uses. fntBL uses the largest window size among the algorithms – a context of up to seven tokens and/or classes – to find the correct class for the token. TNT uses three tags totally, and the MXPOST a context of five words and two tags.

To summarize the effect of the linguistic information included in training, best results can be obtained for all algorithms when lexical information is excluded from both training
Figure 7: The error rate for each classifier when training is performed entirely on the basis of the PoS to predict the phrase labels.

and test. However, if the user would like to use lexical information as well, best results can be achieved when both words and PoS tags are included in training to learn the phrase structure. When training is performed on the basis of words only, the statistical approaches MXPOST and TnT prefer to recognize the phrasal structure entirely, without the prediction of PoS tags, while for TBL, the learning tasks where PoS information is involved either as a type of input or as a class is in average easier than when information on PoS is excluded in training.

Thus, the statistical approaches prefer to learn from large amount of types, while for the transformation-based learner, small number of types is preferable.

5.3 The effect of the size of the training data

As we can see from Figures 4, 5, 6, and 7, accuracy is improved for all systems by increasing the size of the training corpus. The fact that the learning task becomes easier by a larger training corpus is not surprising since as we increase the size of the training data, 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 we decrease the percentage of unknown tokens, as it was shown in Figure 2.

The algorithms show different sensitivity to the size of the training corpus in the various experiments. The maximum entropy approach, MXPOST achieves lowest error rate when the training corpus is large and includes lexical information. The hidden Markov model, TnT, on the other hand, obtains lowest errors when trained on small data sets with lexical information included in training. The transformation-based approach, fntBL, shows most sensitivity to the size of training data when training is performed on words with or without PoS to predict the constituent structure.

The reason for the different sensitivity the algorithms show can be explained by the type of background knowledge that is implemented in the systems. An explanation to why
TNT achieves lowest error rate among the classifiers on small training data where lexical information is included can we find in the system’s analysis of unknown words. As it was shown by Brants (2000) and Megyesi (2001), TNT succeeds well in the annotation of unknown words because of the advanced suffix analysis; TNT checks up to 10 characters for the classification of unknown words, while MXPOST and fntBL uses four characters, only. The success of maximum entropy approach, MXPOST, achieving lowest error rates when the training corpus is large and includes lexical information can we explain by the window size the system use 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. Transformation-based learner, fntBL, on the other hand, does not perform well on small training data and obtains highest error rate in the annotation of unknown words. When entirely PoS categories are used as the basis for learning we eliminate the problem of the analysis of unknown words, thereby making an easier classification for TBL. However, this is not the only reason for TBL’s success. The contextual environment TBL checks for the prediction of the phrasal structure of a particular PoS tag, is largest among the PoS taggers. TBL uses a window size of seven tokens/tags, i.e. a context of up to three preceding and following tokens and/or tags.

Thus, the linguistic information, the size of training data, and the background knowledge for the identification of unknown words and the disambiguation strategies used by the algorithms are all important features for achieving state-of-the-art results for chunking, parsing as it has been proved to be for the PoS tagging task (De Pauw and Daelemans, 2000).

5.4 Evaluation for Real-World Applications

The reader might ask how we can apply the above described results in real-world applications where the system needs both PoS tagged and parsed text. Well, 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 keeping the sentence division, and let the parser annotate the PoS sequences. Lastly, the only thing remaining to do is 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 have to use a tagger for that purpose, the performance of the parser can be expected to decrease. Therefore, an evaluation for real-world applications might be interesting.

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 was used for the real-world evaluation task. The test data was first tagged by a PoS tagger, TNT (trained on the 500k tokens data) since this tagger have been proved to achieve highest accuracy on larger training data for Swedish (Megyesi, 2001). The tagger was trained on the same Second, the words were removed and the POS sequences was labeled with phrase categories by each classifier, i.e. parser, trained on the 500k token data.

The results are shown in Table 5.4. The performance of the PoS tagger is 94.98%. The performance of the parsers are considerably lower as it was predicted. FNTBL achieved
the highest accuracy, followed by MXPOST and TNT, just as it was the case in the fourth experiment.

<table>
<thead>
<tr>
<th>TAGGER</th>
<th>PARSER</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>fntbl</td>
<td>90.83%</td>
<td></td>
</tr>
<tr>
<td>94.98%</td>
<td>MXPOST 88.87%</td>
<td></td>
</tr>
<tr>
<td>TNT</td>
<td>88.19%</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Accuracy (%) is given for each classifier when the test set was first tagged by TNT, then parsed with all three classifiers based on PoS sequences only.

6. Conclusion and Future Work

This paper presented empirical results on the application of PoS taggers to shallow parse Swedish texts. The PoS taggers included in the study was the transformation-based learner fntBL, the maximum entropy approach MXPOST, and the implementation of hidden Markov model, Trigrams’n’Tags (TNT). The goal of the shallow parsers is recognize the constituent structure of the sentence, marking the whole hierarchical structure the token belongs to in the parse tree. The results show that the data-driven, language and tag set independent PoS taggers can be efficiently used to shallow parse texts, given that PoS information only, without the presence of words, is included in the training data.

The classifiers were evaluated from several aspects. The algorithms' sensitivity to several kinds of linguistic information included in the training data has been carefully examined. A particular attention has been directed to various types of input features that the algorithms learn from, such as words, PoS tags, and a combination of both. Also, experiments have been carried out on various amounts of classes that the algorithms 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 algorithms sensitivity to the size of the training data including different linguistic information was investigated.

The results show that for all three systems, best performance can be obtained if the number of token types the algorithms learn from is reduced. It is shown that by only considering the PoS tags, i.e. excluding the lexical information during learning and test, all algorithms obtains above 92% when large training data is used for training.

However, the type of linguistic information, the size of the training data, the size of the contextual environment, and the analysis of the token types are all factors that influence system performance. The results, described in this study, show clear differences between the algorithms' sensitivity to the type of information used in learning and the amount of classes to be learned.

The transformation-based learner, fntBL, obtains best results when training is performed on PoS categories only, due to a large window size the algorithm uses for the disambiguation of known words. However, tbl does not succeed as well as the statistical approaches in the analysis of the unknown tokens. However, it has to be pointed out, that an additional
large lexicon listing all possible classes for a token type, was not used in this study. If
such a lexicon would be included in test, accuracy could be expected to increase, thereby
decreasing the amount of unknown words, as it was reported by Megyesi (2001).

The maximum entropy learner, MXPOST, succeeds best in average when training is
performed on large data sets containing lexical information (i.e. words) of many types.
The success of MXPOST can we also explain by the information bias included on the size
of the context window. MXPOST uses a smaller context than TBL, totally five words and
two tags compared to TBL’s seven words and/or tags. By looking at smaller contextual
environments, in which a particular token can appear, larger amount of examples is created.
The classification task becomes thereby easier.

On the hidden Markov model based TNT outperforms all systems when the size of the
training data is small, including information on the words with or without PoS tags. When
small training data is used, the percentage of unknown tokens is considerably large, making
classification more difficult. Since TNT uses a generous suffix analysis by looking at up to
ten last characters of the tokens (compared to the other systems with a suffix analysis up to
four characters), unknown word prediction becomes more accurate, and classification more
reliable. The reason for that does not succeed as well as MXPOST on larger training data
where the number of unknown tokens is low, is that TNT only uses a context of two tags.

Since the taggers were used with their default setting, using different types of background
knowledge for the analysis of unknown words and the disambiguation of known words, it
is not possible to give a description of the differences between the algorithm bias. Future
work therefore includes the 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 parsers were not optimized for Swedish, it would be necessary to
find out the best combination of feature settings in the 'background knowledge'. Furthermore,
the author’s intention is to include the memory-based learner into the study as soon
as the implementation of the tagger is released.

Also, future research includes the evaluation of the already trained parsers in terms of
precision and recall rates for chunks only, such as base NPs, verb clusters, and PPs, by not
considering the correctness of the phrasal categories on lower and/or higher nodes in the
tree in the evaluation process. In this way, the number of phrases that a token belongs to
would decrease from between 407 and 1,033 tags to three phrase label tags, hence accuracy
can be assumed to be further improved.

Lastly, future work also includes the creation of automatic data-driven detection of
clause boundaries, such as relative clauses and other subordinate clauses in order to be able
to detect even more complex noun phrases, including verb phrases.

Acknowledgements

Thousands of thanks go to my supervisor Rolf Carlson for his encouragement and sup-
port. Also, many thanks to all the researchers who created the PoS taggers used in this
study for making their system freely available for research: Thorsten Brants, Radu Florian
and Grace Ngai, and Adwait Ratnaparkhi. This research was carried out at the Centre
for Speech Technology, a competence center at KTH, supported by VINNOVA and industrial
partners representing Swedish speech technology companies.
References


