Chunk Tagger

Statistical Recognition of Noun Phrases

Wojciech Skut and **Thorsten Brants**
Universität des Saarlandes
Computational Linguistics
D-66041 Saarbrücken, Germany
{skut,brants}@coli.uni-sb.de

In ESSLLI-98 Workshop on Automated Acquisition of Syntax and Parsing, Saarbrücken, 1998

Abstract

We describe a stochastic approach to *partial parsing*, i.e., the recognition of syntactic structures of limited depth. The technique utilises Markov Models, but goes beyond usual bracketing approaches, since it is capable of recognising not only the boundaries, but also the internal structure and syntactic category of simple as well as complex NP's, PP's, AP's and adverbials. We compare tagging accuracy for different applications and encoding schemes.

1 Motivation

The word *chunking* (also *partial* or *shallow parsing*) refers to techniques used for the recognition of relatively simple syntactic structures, such as NPs, PPs, verb complexes etc.

NP chunkers typically rely on fairly simple and efficient processing tools such as finite automata or (in stochastic approaches) Markov Models (MMs). The output consists of structures recognised with a high degree of certainty; such structures are used for further processing.

1.1 Annotation Software

The main motivation for the work reported in this paper was the development of NLP software for creating language resources, especially syntactically annotated corpora (treebanks). A disadvantage of symbolic tools supporting corpus annotation is that they are language specific and have to be developed prior to actual annotation. For English, this is not a problem since there are many such tools, yet for other languages, serious difficulties may arise here.

As an alternative, a bootstrapping approach can be taken in which, after a short phase of purely manual annotation, more and more automatic procedures are implemented using statistical NLP methods. The already annotated sentences serve as training data. This approach is highly independent of the annotation format, which is simply learned from training data.

With these prerequisites, we have developed a stochastic parser (*chunker*) that recognises syntactic structures of limited depth. The tool is language-independent and can be used for parsing unrestricted text provided some minimal amount of annotated data is available.

1.2 Overview

In the following, we describe our stochastic approach to NP chunking based on a generalisation of standard POS tagging techniques (hence the name *chunk tagger*). First, we show how a simple bracketing method can be extended to recognise more complex structures and several types of phrases (sections 2 and 3). Accuracy for different applications and tasks is reported in section 4. In section 5, we discuss the similarities and differences between our approach and related research.

2 Stochastic NP Recognition

The idea of using statistics for NP chunking goes back to Church (1988), who used corpus frequencies to determine the boundaries of simple non-recursive NP's. For each pair of POS tags t_i, t_j, the probability of an NP boundary ('[' or ']') occurring between t_i and t_j is computed. On the basis of these context probabilities, the program inserts the symbols '[' and ']' into sequences of POS tags, yielding output of the following form:

[A/AT former/AP top/NN aide/NN] to/IN [Attorney/NP General/NP Edwin/NP Meese/NP] interceded/VBD to/TO extend/VB [an/AT aircraft/NN company/NN...

The accuracy of this approach is impressive. On the other hand, the task is not too difficult since recursive structures are not recognised. It is interesting whether this simple technique can be used for the recognition of more complex phrases.

2.1 Beyond Simple Bracketing

We have modified Church's approach in a way permitting efficient and reliable recognition of structures of limited depth, including complex prenominal adjectival and participial phrases, postnominal PP's and genitives, appositions, etc. We encode the structure in such a way that it can be recognised by a part-of-speech tagger, so the process runs in time linear to the length of the input string.

The basic idea is simple enough: structures of limited depth are encoded using a finite number of flags. We employ flags standing not just for bracketing, but for structural relations between adjacent words.

Given a sequence of words $\langle w_0, w_1, ...w_n\rangle$, we consider the structural relation r_i holding between w_i and w_{i-1} for $1 \leq i \leq n$. For the recognition of most – even fairly complex – NPs, PPs, and APs, it is sufficient to distinguish the following seven values of r_i which uniquely identify sub-structures of limited depth.

$$r_i = \begin{cases} 0 & \text{if} \quad parent(w_i) = parent(w_{i-1}) \\ + & \text{if} \quad parent(w_i) = parent^2(w_{i-1}) \\ ++ & \text{if} \quad parent(w_i) = parent^3(w_{i-1}) \\ - & \text{if} \quad parent^2(w_i) = parent(w_{i-1}) \\ -- & \text{if} \quad parent^3(w_i) = parent(w_{i-1}) \\ = & \text{if} \quad parent^2(w_i) = parent^2(w_{i-1}) \\ 1 & \quad \text{else} \end{cases}$$

If more than one of the conditions above are met, the first of the corresponding tags in the list is assigned. The depth of structures is limited to 3. For convenience, we give the graphical representation of the structural tags in figure 1. A structure tagged with these symbols is shown in figure 2.

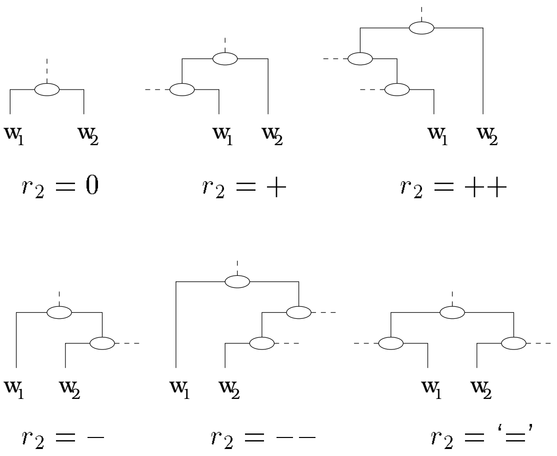


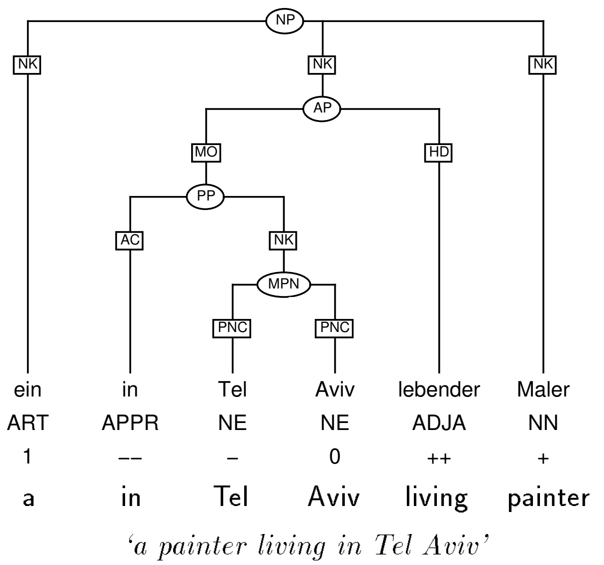
Figure 1: Structural tags r_2 assigned to w_2

Instead of the simple context frequencies used by Church, we employ a generalisation of the standard MM-based POS tagging method. The task of the chunker is to assign the most probable sequence of structural tags $R = \langle r_0, r_1, \ldots, r_n\rangle$ to a sequence of POS tags $T = \langle t_0, t_1, \ldots, t_n\rangle$. This can be done exactly in the same way as the assignment of the optimal POS sequence to a sequence of words in POS tagging, and the task is to calculate

$$\underset{R}{\operatorname{argmax}}\, P(R|T) \tag{1}$$

$$\begin{aligned} &= \underset{R}{\operatorname{argmax}} \frac{P(R) \cdot P(T|R)}{P(T)} \\ &= \underset{R}{\operatorname{argmax}}\, P(R) \cdot P(T|R) \\ &= \underset{R}{\operatorname{argmax}} \prod_{i=1}^{k} P(r_i|r_{i-2}, r_{i-1}) P(t_i|r_i) \end{aligned}$$

Under this perspective, a standard part-of-



AC = adpositional case marker, HD = head, MO = modifier, MPN = multi-token proper noun, NK = noun phrase kernel, PNC = proper noun constituent

Figure 2: Encoding of a sample structure

speech tagger can be trained on a syntactically annotated corpus with structures converted into structural tags (the r_i's). However, in this case the corresponding Markov Model has only 7 tags (the possible values of r_i), which is obviously too coarse-grained. The precision of the tagger is rather disappointing: only about 77% of all structures are recognised correctly.

To cope with this problem, we enrich the MM state with information about the POS tag t_i assigned to w_i. Now we can define *structural tags* as pairs $S_i = \langle r_i, t_i \rangle$. Such tags constitute a finite alphabet of symbols describing structures of depth ≤ 3.

The tagger's task is thus to assign the most probable sequence of structural tags $S = \langle S_0, S_1, ..., S_n \rangle$ to a sequence of part-of-speech tags $T = \langle t_0, t_1, ..., t_n \rangle$, hence

$$\operatorname*{argmax}_S P(S|T) \quad (2)$$

$$= \operatorname*{argmax}_S P(S) \cdot P(T|S)$$

The part-of-speech tags are encoded in the structural tag (the t_i dimension), so S uniquely determines T. Therefore, we have $P(t_i|S_i) = 1$ if $S_i = \langle r_i, t_i \rangle$ and 0 otherwise, which simplifies calculations.

The contexts are smoothed by linear interpolation of unigrams, bigrams, and trigrams. Their weights are calculated by deleted interpolation.

3 Phrasal Categories

A simple extension of the chunk tagger can assign phrasal categories in addition to structures. We enrich the state S_i of the Markov Model with information about the category c_i of the node immediately dominating word w_i. Thus S_i becomes a triple $\langle r_i, t_i, c_i \rangle$. For example, the adjective *lebender* in figure 2 is assigned the tag $\langle ++, ADJA, AP \rangle$. This extension also slightly improves the recognition of structures, cf. section 4.

Further precision gain can be achieved if we also add some information g_i about the category of the grandparent node. However, only few symbols can be used to encode this dimension. Otherwise, the tagset (all $S_i = \langle r_i, t_i, c_i, g_i \rangle$) becomes too large. We achieved the best results with just three flags A, N and C, which indicate that $parent^2(w_i)$ is an AP, an NP/PP and a coordinated constituent, respectively. In this format, the word *Aviv* in figure 2 is assigned the tag $\langle 0, NE, MPN, N \rangle$.

4 Applications and Results

In this section, we compare results achieved for different applications and types of structures. We use the dependency-oriented NEGRA treebank (Skut et al., 1997) as training data. The current size of the corpus is 12,000 sentences, or 210,000 tokens. All of these sentences have been annotated without the help of the chunk tagger.

The annotation scheme distinguishes 24 phrasal categories. The POS tagset (Thielen and Schiller, 1995) consists of 54 tags. For tagging purposes, it has been adjusted by merging tags irrelevant to the chunking task and adding simple morphological and lexical information. Due to this adjustment, 1.5% more words are assigned the correct structural tag.

Structures are encoded according to the method presented in section 3. We vary the number of tag dimensions (1 – 4).

The results given in the following sections have been computed by spliting the corpus into disjoint training and test parts (90% and 10%, respectively). This procedure was repeated ten times, and the results were averaged. The accuracy measures employed are explained as follows.

tags: the percentage of structural tags with the correct value of the r_i attribute,

bracketing: the percentage of correctly recognised nodes,

labelled bracketing: the percentage of nodes recognised correctly including their syntactic category,

top-level chunks: the percentage of correctly parsed "maximal" chunks, i.e., phrases not contained in a larger chunk of depth ≤ 3.

We present figures concerning the *precision* of the chunker. *Recall* is slightly lower for all applications (0.5% – 1.5%).

4.1 Corpus Annotation

As we already mentioned, the primary application of the chunk tagger is corpus (treebank) annotation. The treebank is being created in an interactive annotation mode: automatic and manual annotation steps are closely interleaved to ensure optimal control of the predictions made automatically (for a precise description of this interactive approach to treebank annotation see (Skut et al., 1997)).

As for the chunker, the interactive annotation mode means that the annotator specifies the boundaries of a complex NP or PP, and the tool recognises its category and internal structure. Note that the disambiguation of PP attachment is significantly easier than in the general case. Correct structural tags are assigned to more than 95% of all words, which means that approx. 89% of all chunks (NP's, PP's, AP's) are assigned the correct syntactic structure.

Precise results for different chunk encoding methods are given in table 1. The training corpus was created by extracting all NPs, PPs and APs occurring in the NEGRA treebank (34,000 chunks, 130,000 tokens). In other words, the program had to perform the annotator's task: find a labelled structure that spans a given sequence of words.

Table 1: Precision of the chunk tagger in the interactive annotation mode for different chunk encoding methods. The symbols in brackets denote: r structural relation (7 values), t POS tag (54 values), c parent node category (24 values), g grandparent node category (3 values).

dimensions	tags (r_i)	brack.	l. brack.
$\langle r \rangle$	87.8%	76.6%	–
$\langle r, c, g \rangle$	92.4%	83.9%	78.1%
$\langle r, t \rangle$	94.7%	88.3%	–
$\langle r, t, c \rangle$	94.9%	88.7%	84.7%
$\langle r, t, c, g \rangle$	95.1%	89.2%	85.2%

It can be seen from the table that part-of-speech information (t) is much more relevant than information about phrasal categories (c and g). The latter also enhances the performance of the tagger, but the improvement achieved is rather modest.

The tagset size and average ambiguity for the five encoding schemes are shown in table 2.

Table 2: Tagset sizes and ambiguity.

dimensions	# tags	tags per word
$\langle r \rangle$	7	4.5
$\langle r, c, g \rangle$	125	24.9
$\langle r, t \rangle$	251	4.5
$\langle r, t, c \rangle$	775	18.7
$\langle r, t, c, g \rangle$	996	24.9

With a unigram model, i.e. choosing the

most probable tag without looking at the context, tag assignment precision is only 45.8% (for $S = \langle r, t, c, g \rangle$). The precision of a bigram model is 94.3%. Thus the difference to the trigram model is very small, which obviously results from the fairly large amount of information encoded in a single chunk tag (structural relation, POS tag and phrasal category), so that a trigram context does not contain much more information than a bigram one.

4.2 Tagging the Penn Treebank

In order to better evaluate the performance of the chunk tagger, we applied it to a fragment of the Penn Treebank. As in the evaluation reported in the previous section, the training corpus consisted of all NP's, PP's and AP's occurring in the Treebank fragment. In the table below, the results are contrasted with those of chunk tagging the NEGRA corpus.

Table 3: Precision for different corpora in the interactive annotation mode (Penn Treebank and NEGRA Corpus). Information about external phrase boundaries is supplied by the annotator.

corpus	PTB	NEGRA
# sentences	10,000	12,000
# top-level chunks	33,808	33,787
# phrasal nodes	88,083	56,110
tags (r_i)	93.0%	95.1%
bracketing	91.1%	89.2%
lab. bracketing	86.7%	85.2%
top-level chunks	81.3%	88.8%

The figures show that the top-level chunk recognition rate is significantly better for the NEGRA corpus data. The difference seems to arise from the higher structural complexity of the Penn Treebank fragment, where a chunk on average contains 2.56 phrasal nodes (as opposed to 1.65 in the NEGRA corpus, which does not contain unary projections).

4.3 Other Applications

The chunk tagger can also be used as a standalone application, i.e., for the recognition of simple structures in text. This task is obviously more difficult since all phrase boundaries have to be recognised by the tagger. As a result, precision drops significantly, cf. table 4.

Table 4: Precision of the chunk tagger with PP/NP/adverb attachment. No pre-editing by a human annotator.

measure	correct
structural tags (r_i)	90.9%
bracketing	75.4%
labelled bracketing	72.6%
top-level chunks	71.1%

To find the main sources of errors, we examined the results and found that erroneous output mostly originated from wrong *PP attachment*. Furthermore, many errors were due to coordination and *focus adverbs* (e.g., *nur* 'only', *auch* 'also', etc.), which introduce a high ambiguity potential.

Since the disambiguation of such attachments involves lexical and even world knowledge, PP and focus adverb attachment cannot be recognised in a satisfactory way by a MM-based tagger operating on POS tags. Thus the best strategy is to postpone attaching PP's and adverbs, and make the chunk tagger recognise the prenominal part of NP's and PP's only. With this modification, precision increases to more than 95%. Exact results are given in table 5.

If we ignore errors concerning the internal structure of the chunks (i.e., we measure only the recognition of external boundaries, which corresponds to the precision measurement in some other approaches), 93.4% of all chunks are assigned the correct part of the input string.

4.4 Size of the Training Corpus

An important advantage of the chunker is that it is independent of theory-internal representations and can be used to recognise structures of

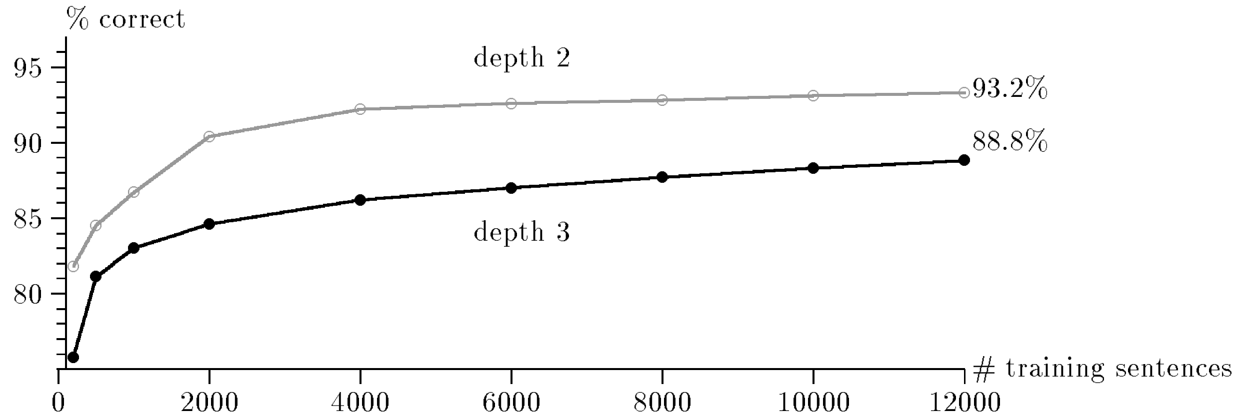


Figure 3: Precision as percentage of correctly recognised top-level chunks of depth 2 and 3, shown for different numbers of training sentences.

Table 5: Precision of the chunk tagger *without* PP/NP/adverb attachment. No pre-editing by a human annotator.

measure	correct
structural tags (r_i)	95.5%
bracketing	89.3%
labelled bracketing	86.2%
top-level chunks	89.0%

any language. Of course, the availability of a training corpus is a prerequisite. Now we shall see how much data is necessary to achieve reliable results.

Figure 3 shows precision (measured as the percentage of top-level chunks recognised correctly) for the interactive annotation mode. We consider two encoding schemes. The *depth 3* scheme is the one described in section 2.1, which uses all the 7 possible values of the r_i dimension. The *depth 2* scheme is its slightly simplified version in which r_i can take only four values: **1, 0, +, -**, so that only depth-two trees are recognised by the chunker.

While for the depth 3 encoding a training corpus of 1000–2000 sentences is needed, the simpler encoding requires only about 500 sentences. Thus the chunk tagger can be successfully used in treebank annotation at quite an early stage, with only a few hundred annotated sentences available.

5 Related Work

In section 2, we sketched the simple bracketing technique described by Church (1988), which provided motivation for our chunking method. As far as other approaches are concerned, our work is most closely related to that of Joshi and Srinivas (1994), who use Markov Models in a preprocessing step to reduce the number of tree segments (called *supertags*) that can be assigned to a word in a lexicalised Tree Adjoining Grammar. This approach makes parsing more efficient, but it needs a large training corpus, has to fight a large amount of ambiguity and needs a subsequent parsing step (also see (Srinivas, 1996) for the use of explanation-based learning for this purpose).

Symbolic NP chunkers usually rely on finite automata and/or pattern matching, cf. (Koskenniemi, 1990), (Aït-Mokhtar and Chanod, 1997). Abney (1996) presents a partial parsing technique based on cascaded finite automata. Voutilainen and Padró (1997) describe a POS tagger and shallow parser combining symbolic and stochastic processing via *relaxation labelling*.

The precision of the abovementioned approaches is often measured by the number of correct labels assigned to words. The figures range from 85% to 98%. Our results (89% – 95%) fit into this interval, yet it should be kept in mind that the coverage of the approaches and the precision measuring methods are often only roughly comparable. For instance, several shallow parsing methods are restricted to POS tagging and grammatical function labelling without explicitly specifying attachments and phrase boundaries. Furthermore, the notion of 'phrase' varies in these investigations, and usually these studies concentrate on simple, non-recursive structures. By contrast, our chunker is capable of recognizing complex, even recursive, NPs, PPs, and APs.

Compared to the symbolic techniques, an important advantage of the stochastic approach taken in our project is its independence of external lexical resources. As a result, the chunker trained with the POS-tags and structures of the current corpus is fairly domain-independent. Of course, our tool would benefit from the use of lexical knowledge; this issue has to be addressed in the near future.

Since our approach is restricted to a small number of structurally different tags, it has the great advantage of requiring only a small amount of training data (cf. section 4) and the recognition of these phrases is of high accuracy.

6 Conclusion

We have presented a stochastic partial parser (*chunker*) that recognises the boundaries, internal structure and syntactic category of simple as well as fairly complex NP's, PP's and AP's. The chunker is a straightforward application of a stochastic part-of-speech tagger. We use it to model a mapping from lexical categories to syntactic structures. The type of the structural encoding is crucial in this approach, and better encodings increase the accuracy of structural assignment. The use of Markov Model processing techniques guarantees that the process runs in time linear to the length of the input string.

References

[Abney1996] Steven Abney. 1996. Partial parsing via finite-state cascades. In *Proceedings of the ESSLLI'96 Robust Parsing Workshop*, Prague, Czech Republic.

[Aït-Mokhtar and Chanod1997] S. Aït-Mokhtar and J. Chanod. 1997. Incremental finite state parsing. In *Proceedings of ANLP-97*, Washington, DC.

[Church1988] Kenneth Ward Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. Second Conference on Applied Natural Language Processing*, pages 136–143, Austin, Texas, USA.

[Joshi and Srinivas1994] A. K. Joshi and B. Srinivas. 1994. Disambiguation of super parts of speech (or supertags). In *Proceedings COLING 94*, Kyoto, Japan.

[Koskenniemi1990] K. Koskenniemi. 1990. Finite-state Parsing and disambiguation. In *COLING-90*, Helsinki.

[Skut et al.1997] Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of ANLP-97*, Washington, DC.

[Srinivas1996] B. Srinivas. 1996. Explanation-based learning and finite state transducers: Applications to parsing lexicalised tree adjoining grammars. In *Proceedings of the Workshop on Finite-State Models of Language, ECAI 96*, Budapest.

[Thielen and Schiller1995] Christine Thielen and Anne Schiller. 1995. Ein kleines und erweitertes Tagset fürs Deutsche. In *Tagungsberichte des Arbeitstreffens Lexikon + Text 17./18. Februar 1994, Schloß Hohentübingen. Lexicographica Series Maior*, Tübingen. Niemeyer.

[Voutilainen and Padró1997] A. Voutilainen and L. Padró. 1997. Developing a hybrid NP parser. In *Proceedings of ANLP-97*, Washington, DC.