Robust German Noun Chunking With a Probabilistic Context-Free Grammar

Helmut Schmid and Sabine Schulte im Walde[*]
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
Azenbergstraße 12, 70174 Stuttgart, Germany
{schmid,schulte}@ims.uni-stuttgart.de

Abstract

We present a noun chunker for German which is based on a head-lexicalised probabilistic context-free grammar. A manually developed grammar was semi-automatically extended with robustness rules in order to allow parsing of unrestricted text. The model parameters were learned from unlabelled training data by a probabilistic context-free parser. For extracting noun chunks, the parser generates all possible noun chunk analyses, scores them with a novel algorithm which maximizes the best chunk sequence criterion, and chooses the most probable chunk sequence. An evaluation of the chunker on 2,140 hand-annotated noun chunks yielded 92% recall and 93% precision.

1 Introduction

A *noun chunker* marks the noun chunks in a sentence as in the following example:

(Wirtschaftsbosse)	mit	(zweifelhaftem	Ruf)		
economy chefs	with	doubtable	reputation		
sind	an	(der in	(Engpässen)	angewandten	Führung)
are	in	the in	bottlenecks	applied	guidance
(des	Landes)	beteiligt.			
of the	country	involved.			

> 'Leading economists with doubtable reputations are involved in guiding the country in times of bottlenecks.'

A tool which identifies noun chunks is useful for term extraction (most technical terms are nouns or complex noun groups), for lexicographic purposes (see (Tapanainen and Järvinen, 1998) on syntactically organised concordancing), and as index terms for information retrieval. Chunkers may also mark other types of chunks like verb groups, adverbial phrases or adjectival phrases.

Several methods have been developed for noun chunking. Church's noun phrase tagger (Church, 1988), one of the first noun chunkers, was based on a Hidden Markov Model (HMM) similar to those used for part-of-speech tagging. Another HMM-based approach has been developed by Mats Rooth (Rooth, 1992). It integrates two HMMs; one of them models noun chunks internally, the other models the context of noun chunks. Abney's cascaded finite-state parser (Abney, 1996) also contains a processing step which recognises noun chunks and other types of chunks. Ramshaw and Marcus (Ramshaw and Marcus, 1995) successfully applied Eric Brill's transformation-based learning method to the chunking problem. Voutilainen's NPtool (Voutilainen, 1993) is based on his constraint-grammar system. Finally, Brants (Brants, 1999) described a German chunker which was implemented with cascaded Markov Models.

In this paper, a probabilistic context-free parser is applied to the noun chunking task. The German grammar used in the experiments was semi-automatically extended with robustness rules in order to be able to process arbitrary input. The grammar parameters were trained on unlabelled data. A novel algorithm is used for noun chunk extraction. It maximises the probability of the chunk set.

The following section introduces the grammar framework, followed by a description of the chunking algorithm in section 3, and the experiments and their evaluation in section 4.

2 The Grammar

The German grammar is a head-lexicalised probabilistic context-free grammar. Section 2.1 defines probabilistic context-free grammars and their head-lexicalised refinement. Section 2.2 introduces our grammar architecture, focusing on noun chunks. The robustness rules for the chunker are described in section 2.3.

2.1 (Head-Lexicalised) Probabilistic Context-Free Grammars

A *probabilistic context-free grammar* (PCFG) is a context-free grammar which assigns a probability P to each context-free grammar rule in the rule set R. The probability of a parse tree T is defined as $\prod_{r \in R} P(r)^{|r|}$, where $|r|$ is the number of times rule r was applied to build T. The parameters of

[*] Thanks to Mats Rooth and Uli Heid for many helpful comments.

PCFGs can be learned from unparsed corpora using the Inside-Outside algorithm (Lari and Young, 1990).

Head-lexicalised probabilistic context-free grammars (H-L PCFG) (Carroll and Rooth, 1998) extend the PCFG approach by incorporating information about the lexical head of constituents into the probabilistic model.[1] Each node in a parse of a H-L PCFG is labelled with a category and the lexical head of the category. A H-L PCFG rule looks like a PCFG rule in which one of the daughters has been marked as the head. The rule probabilities $P_{rule}(C \rightarrow \alpha | C)$ are replaced by lexicalised rule probabilities $P_{rule}(C \rightarrow \alpha | C, h)$ where h is the lexical head of the mother constituent C. The probability of a rule therefore depends not only on the category of the mother node, but also on its lexical head. Assume that the grammar has two rules `VP` $\rightarrow$ `V NP` and `VP` $\rightarrow$ `V`. Then the transitive verb *buy* should have a higher probability for the former rule whereas the latter rule should be more likely for intransitive verbs like *sleep*. H-L PCFGs incorporate another type of parameters called lexical choice probabilities. The lexical choice probability $P_{choice}(h_d | C_d, C_m, h_m)$ represents the probability that a node of category C_d with a mother node of category C_m and lexical head h_m bears the lexical head h_d. The probability of a parse tree is obtained by multiplying lexicalised rule probabilities and lexical choice probabilities for all nodes. Since it is possible to transform H-L PCFGs into PCFGs, the PCFG algorithms are applicable to H-L PCFGs.

2.2 Noun Chunks in the German Grammar

Currently, the German grammar contains 4,619 rules and covers 92% of our 15 million words of verb final and relative clauses[2]. The structural noun chunk concept in the grammar is defined according to Abney's chunk style (Abney, 1991) who describes chunks as syntactic units which correspond in some way to prosodic patterns, containing a content word surrounded by some function word(s): all words from the beginning of the noun phrase to the head noun are included.[3] The different kinds of noun chunks covered by our grammar are listed below and illustrated with examples:

- a combination of a non-obligatory determiner, optional adjectives or cardinals and the noun itself:

  (1) *eine gute Idee*
  a good idea

  (2) *vielen Menschen*
  for many people

  (3) *deren künstliche Stimme*
  whose artificial voice

  (4) *elf Ladungen*
  eleven cargos

  (5) *Wasser*
  water

  and prepositional phrases where the definite article of the embedded noun chunk is morphologically combined with a preposition, so the pure noun chunk could not be separated:

  (6) *zum Schluss*
  at the end

- personal pronouns: *ich* (I), *mir* (me)
- reflexive pronouns: *mich* (myself), *sich* (himself/herself/itself)
- possessive pronouns:

  (7) *Meins* ist sauber.
  Mine is clean.

- demonstrative pronouns:

  (8) *Jener* fährt viel schneller.
  That one goes much faster.

- indefinite pronouns:

  (9) *Einige* sind durchgefallen.
  Some failed.

- relative pronouns:

  (10) Ich mag Menschen, *die* ehrlich sind.
  I like people who are honest.

- nominalised adjectives: *Wichtigem* (important things)
- proper names: *Christoph Kolumbus*
- a noun chunk refined by a proper name:

  (11) *der Eroberer Christoph Kolumbus*
  the conquerer Christoph Columbus

- cardinals indicating a year:

  (12) Ich begann *1996*.
  I started 1996.

The chunks may be recursive in case they appear as complement of an adjectival phrase, as in *(der (im Regen) wartende Sohn)* (the son who was waiting in the rain).

Noun chunks have features for case, without further agreement features for nouns and verbs. The case is constrained by the function of the noun chunk, as verbal or adjectival complement with nominative, accusative, dative or genitive case, as modifier with genitive case, or as part of a prepositional

[1] Other types of lexicalised PCFGs have been described in (Charniak, 1997), (Collins, 1997), (Goodman, 1997), (Chelba and Jelinek, 1998) and (Eisner and Satta, 1999).

[2] The restricted corpora were extracted automatically from the *Huge German Corpus (HGC)*, a collection of German newspapers as well as specialised magazines for industry, law, computer science.

[3] As you will see below, there is one exception, noun chunks refined by a proper name, which end with the name instead of the head noun.

phrase (also in the special case representing a prepositional phrase itself) with accusative or dative case.

Both structure and case of noun phrases may be ambiguous and have to be disambiguated:

- ambiguity concerning structure:
  diesen (this) is disregarding the context a demonstrative pronoun ambiguous between representing a standalone noun chunk (cf. example (8)) or a determiner within a noun chunk (cf. example (2))
- ambiguity concerning case:
  die Beiträge (the contributions) is disregarding the context ambiguous between nominative and accusative case

The disambiguation is learned during grammar training, since the lexicalised rule probabilities as well as the lexical choice probabilities tend to enforce the correct structure and case information. Considering the above examples, the trained grammar should be able to parse *diesen Krieg* (this war) as one noun chunk instead of two (with *diesen* representing a standalone noun chunk) because of (i) the preferred use of demonstrative pronouns as determiners ($\rightarrow$ lexicalised rule probabilities), and (ii) the lexical coherence between the two words ($\rightarrow$ lexical choice probabilities); in a sentence like *er zahlte die Beiträge* (he paid the contributions) the accusative case of the latter noun chunk should be identified because of the lexical coherence between the verb *zahlen* (pay) and the lexical head of the subcategorised noun phrase *Beitrag* (contribution) as related direct object head ($\rightarrow$ lexical choice probabilities).

2.3 Robustness Rules

The German grammar covers over 90% of the clauses of our verb final and relative clause corpora. This is sufficient for the extraction of lexical information, e.g. the subcategorisation of verbs (see (Beil et al., 1999)). For chunking, however, it is usually necessary to analyse *all* sentences. Therefore, the grammar was augmented with a set of robustness rules. Three types of robustness rules have been considered, namely *unigram* rules, *bigram* rules and *trigram* rules.

Unigram rules are rules of the form `X` $\rightarrow$ `YP X`, where `YP` is a grammatical category and `X` is a new category. If such a rule is added for each grammar category[4], the coverage is 100% because the grammar is then able to generate any sequence of category labels. In practice, some of the rules can be omitted while still retaining full coverage: e.g. the rule `X` $\rightarrow$ `ADV X` is not necessary if the grammar already contains the rules `ADVP` $\rightarrow$ `ADV` and `X` $\rightarrow$ `ADVP X`. Unigram rules are insensitive to their context so that all permutations of the categories which are generated by the X chain have the same probability.

The second type of robustness rules, called *trigram rules* (Carroll and Rooth, 1998) is more context sensitive. Trigram rules have the form `X:Y` $\rightarrow$ `Y Y:Z` where `X, Y, Z` are categories and `X:Y` and `Y:Z` are new categories. Trigram rules choose the next category on the basis of the two preceding categories. Therefore the number of rules grows as the number of categories raises to the third power. For example, 125,000 trigram rules are needed to generate 50 different categories in arbitrary order.

Since unigram rules are context insensitive and trigram rules are too numerous, a third type of robustness rules, called *bigram rules*, was developed. A bigram rule actually consists of two rules, a rule of the form `:Y` $\rightarrow$ `Y Y:` which generates the constituent `Y` deterministically, and a rule `Y:` $\rightarrow$ `:Z` which selects the next constituent `Z` based on the current one. Given n categories, we obtain n rules of the first form and n^2 rules of the second form. Even when categories which directly project to some other category were omitted in the generation of the bigram rules for our German grammar, the number of rules was still fairly large. Hence we generalised some of the grammatical categories by adding additional chain rules. For example, the prepositional phrase categories `PP.Akk:an, PP.Akk:auf, PP.Akk:gegen` etc. were generalised to `PPX` by adding the rules `PPX` $\rightarrow$ `PP.Akk:an` etc. Instead of $n+1$ bigram rules for each of the 23 prepositional categories, we now obtained only $n+2$ rules with the new category `PPX`. Altogether, 3,332 robustness rules were added.

3 Chunking

A head-lexicalised probabilistic context-free parser, called *LoPar* (Schmid, 1999), was used for parsing. The functionality of LoPar encompasses purely symbolic parsing as well as Viterbi parsing, inside-outside computation, POS tagging, chunking and training with PCFGs as well as H-L PCFGs. Because of the large number of parameters in particular of H-L PCFGs, the parser smoothes the probability distributions in order to avoid zero probabilities. The absolute discounting method (Ney et al., 1994) was adapted to fractional counts for this purpose. LoPar also supports lemmatisation of the lexical heads of a H-L PCFG. The input to the parser consists of ambiguously tagged words. The tags are provided by a German morphological analyser (Schiller and Stöckert, 1995).

The best chunk set of a sentence is defined as the set of chunks (with category, start and end position)

[4]Also needed are two rules which start and terminate the "X chain". We used the rules `TOP` $\rightarrow$ `START X` and `X` $\rightarrow$ `END`. `START` and `END` expand to SGML tags which mark the beginning and the end of a sentence, respectively.

for which the sum of the probabilities of all parses which contain exactly that chunk set is maximal. The chunk set of the most likely parse (i.e. Viterbi parse) is not necessarily the best chunk set according to this definition, as the following PCFG shows.

S → A	0.6	B → x	1.0
S → B	0.4	C → x	1.0
A → C	0.5	D → x	1.0
A → D	0.5		

This grammar generates the three parse trees `(S (A (C x)))`, `(S (A (D x)))`, and `(S (B x))`. The parse tree probabilities are 0.3, 0.3 and 0.4, respectively. The last parse is therefore the Viterbi parse of `x`. Now assume that {`A,B`} is the set of chunk categories. The most likely chunk set is then {`(A,0,1)`} because the sum of the probabilities of all parses which contain `A` is 0.6, whereas the sum over the probabilities of all parses containing `B` is only 0.4.

`computeChunks` is a slightly simplified pseudo-code version of the actual chunking algorithm:

computeChunks(G, p_{rule})
Initialize float array $p[G_V]$
Initialize chunk set array $chunks[G_V]$
for each vertex v in G_V in bottom-up order do
  if v is an or-node then
    Initialize float array $prob[chunks[d(v)]]$ to 0
    for each daughter $u \in d(v)$ do
      $prob[chunks[u]] \leftarrow prob[chunks[u]] + p[u]$
    $chunks[v] \leftarrow argmax_c\ prob[c]$
    $p[v] \leftarrow prob[chunks[v]]$
  else
    $p[v] \leftarrow p_{rule}[rule(v)] \prod_{u \in d(v)} p[u]$
    $chunks[v] \leftarrow \bigcup_{u \in d(v)} chunks[u]$
    if v is labelled with a chunk category C then
      $chunks[v] \leftarrow chunks[v] \cup \{(C, start(v), end(v))\}$
return $chunks[root(G)]$

`computeChunks` takes two arguments. The first argument is a parse forest G which is represented as an and-or-graph. G_V is the set of vertices. The second argument is the rule probability vector. d is a function which returns the daughters of a vertex. The algorithm computes the best chunk set $chunks[v]$ and the corresponding probability $p[v]$ for all vertices v in bottom-up order. $chunks[d(v)]$ returns the set of chunk sets of the daughter nodes of vertex v. $rule(v)$ returns the rule which created v and is only defined for and-nodes. $start(v)$ and $end(v)$ return the start and end position of the constituent represented by v.

The chunking algorithm was experimentally compared with chunk extraction from Viterbi parses. In 35 out of 41 evaluation runs with different parameter settings[5], the f-score of the chunking algorithm was better than that of the Viterbi algorithm. The average f-score of the chunking algorithm was 84.7 % compared to 84.0 % for the Viterbi algorithm.

[5]The runs differed wrt. training strategy and number of iterations. See section 4 for details.

4 Experiments

We performed two main chunking experiments. Initially, the parser trained the chunk grammar based on the restricted grammar described in section 2 according to four different training strategies. A preferred training strategy was then applied to investigate the potential of grammar refinement and extended training data.

4.1 Training

In the first experiment, the chunker version of the grammar was trained on a corpus comprising a 1 million word subcorpus of relative clauses, a 1 million word subcorpus of verb final clauses and 2 million words of consecutive text. All data had been extracted from the Huge German Corpus. The test data used for the later evaluation was not included in the training corpus.

For training strategy 1, the chunker grammar was first trained on the whole corpus in unlexicalised mode, i.e. like a PCFG. The parameters were reestimated once in the middle and once at the end of the corpus. In the next step, the grammar was lexicalised, i.e. the parser computed the parse probabilities with the unlexicalised model, but extracted frequencies for the lexicalised model. These frequencies were summed over the whole corpus. Three more iterations on the whole corpus followed in which the parameters of the lexicalised model were reestimated.

The parameters of the unlexicalised chunker grammar were initialised in the following way: a frequency of 7500 was assigned to all original grammar rules and 0 to the majority of robustness rules. The parameters were then estimated on the basis of these frequencies. Because of the smoothing, the probabilities of the robustness rules were small but not zero.

For training strategy 2, the chunker rules were initialised with frequencies from a grammar without robustness rule extensions, which had been trained unlexicalised on a 4 million subcorpus of verb final clauses and a 4 million word subcorpus of relative clauses.

Training strategy 3 again set the frequency of the original rules to 7500 and of the robustness rules to 0. The parser trained with three unlexicalised iterations over the whole training corpus, reestimating the parameters only at the end of the corpus, in order to find out whether the lexicalised probabilistic parser had been better than the fully trained unlexicalised parser on the task of chunk parsing. Training strategy 4 repeated this procedure, but with initialising the chunker frequencies on basis of a trained grammar.

For each training strategy, further iterations were added until the precision and recall values ceased to improve.

For the second part of the experiments, the base grammar was extended with a few simple verb-first and verb-second clause rules. Strategy 4 was applied for training the chunker

(A) on the same training corpus as before, i.e. 2 million words of relative and verb final clauses, and 2 million words of unrestricted corpus data from the HGC,

(B) on a training corpus consisting of 10 million words of unrestricted corpus data from the HGC.

4.2 Evaluation

The evaluation of the chunker was carried out on noun chunks from 378 unrestricted sentences from the German newspaper *Frankfurter Allgemeine Zeitung (FAZ)*. Two persons independently annotated all noun chunks in the corpus –a total of 2,140 noun chunks–, according to the noun chunk definition in section 2.2, without considering grammar coverage, i.e. noun chunks not actually covered by the grammar (e.g. noun chunk ellipsis such as *die kleinen* $[\ ]_N$) were annotated as such. As labels, we used the identifier `NC` plus case information: `NC.Nom, NC.Acc, NC.Dat, NC.Gen`. In addition, we included identifiers for prepositional phrases where the preposition is morphologically merged with the definite article, (cf. example (6)), also including case information: `PNC.Acc, PNC.Dat`.

For each training strategy described in section 4.1 we evaluated the chunker before the training process and after each training iteration: the model in its current training state parsed the test sentences and extracted the most probable chunk sequence as defined in section 3. We then compared the extracted noun chunks with the hand-annotated data, according to

- the range of the chunks, i.e. did the chunker find a chunk at all?
- the range and the identifier of the chunks, i.e. did the chunker find a chunk and identify the correct syntactic category and case?

Figures 1 and 2 display the results of the evaluation in the first experiment,[6] according to noun chunk range only and according to noun chunk range, syntactic category and case, respectively. Bold font highlights the best versions.

Training strategy 2 with two iterations of lexicalised training produced the best f-scores for noun chunk boundary recognition if unlexicalised parsing was done. The respective precision and recall values were 93.06% and 92.19%. For recognising noun chunks with range, category and case, the best chunker version was created by training strategy 4, after five iterations of unlexicalised training; precision and recall values were 79.28% and 76.75%, respectively.

From the experimental results, we can conclude that:

1. initialisation of the chunker grammar frequencies on the basis of a trained grammar improves the untrained version of the chunker, but the difference vanishes in the training process
2. unlexicalised parsing is sufficient for noun chunk extraction; for extraction of chunks with case information, unlexicalised training turned out to be even more successful than a combination with lexicalised training

Figures 3 and 4 display the results of the evaluation concerning the second experiment, compared to the initial values from the first experiment.

Extending the base grammar and the training corpus slightly increased precision and recall values for recognising noun chunks according to range only. The main improvement was in noun chunk recognition according to range, category and case: precision and recall values increased to 83.88% and 83.21%, respectively.

4.3 Failure Analysis

A comparison of the parsed noun chunks with the annotated data showed that failure in detecting a noun chunk was mainly caused by proper names, for example *Netanjahu*, abbreviations like *OSZE*, or composita like *South China Morning Post*. The diversity of proper names makes it difficult for the chunker to learn them properly. On the one hand, the lexical information for proper names is unreliable because many proper names were not recognised as such. On the other hand, most proper names are too rare to learn reliable statistics for them.
Minor mistakes were caused by (a) articles which are morphologically identical to noun chunks consisting of a pronoun, for example *den Rentnern* (the pensioners$_{dat}$) was analysed as two noun chunks, *den* (demonstrative pronoun) and *Rentnern*, (b) capital letter confusion: since German nouns typically start with capital letters, sentence beginnings are wrongly interpreted as nouns, for example *Würden* as the conditional of the auxiliary *werden* (to become) is interpreted as the dative case of *Würde* (dignity), (c) noun chunk internal punctuation as in *seine ‘ Partner ’* (his ‘ partners ’).

Failure in assigning the correct syntactic category and case to a noun chunk was mainly caused by (a) assigning accusative case instead of nominative case, and (b) assigning dative case or nomina-

[6] The lexicalised chunker versions obtained by strategy 2 were also utilised for parsing the test sentences unlexicalised.

	Strategy 1		Strategy 2		–parsed unlex–		Strategy 3		Strategy 4	
	prec	rec	prec	rec	prec	rec	prec	rec	prec	rec
untrained	83.63%	83.63%	90.22%	90.18%	90.22%	90.18%	83.63%	83.63%	90.22%	90.18%
unlex1	**91.33%**	89.62%	**92.84%**	**91.58%**	92.84%	91.58%	91.33%	89.62%	92.84%	**91.58%**
unlex2							92.55%	90.04%	93.01%	91.49%
unlex3							**92.78%**	**90.22%**	92.95%	91.25%
unlex4									93.09%	90.79%
unlex5									**93.29%**	90.32%
lex0	89.13%	**89.71%**	90.12%	90.41%	93.01%	91.49%				
lex1	88.37%	89.52%	88.97%	89.76%	93.02%	91.67%				
lex2	88.25%	89.57%	89.79%	90.46%	**93.06%**	**92.19%**				
lex3	88.17%	89.62%	89.42%	90.13%	93.05%	92.05%				

Figure 1: Comparing training strategies: noun chunk evaluation according to range only

	Strategy 1		Strategy 2		–parsed unlex–		Strategy 3		Strategy 4	
	prec	rec	prec	rec	prec	rec	prec	rec	prec	rec
untrained	63.52%	63.52%	72.02%	71.98%	72.02%	71.98%	63.52%	63.52%	72.02%	71.98%
unlex1	**74.50%**	73.11%	**75.87%**	74.84%	75.87%	74.84%	74.50%	73.11%	75.87%	74.84%
unlex2							76.88%	74.79%	78.27%	76.99%
unlex3							**77.97%**	**75.82%**	78.70%	**77.27%**
unlex4									78.80%	76.85%
unlex5									**79.28%**	76.75%
lex0	73.68%	73.15%	75.10%	75.35%	**78.27%**	**76.99%**				
lex1	72.02%	72.97%	74.69%	75.35%	77.27%	76.15%				
lex2	72.76%	**73.85%**	75.26%	**75.82%**	77.48%	76.75%				
lex3	71.97%	73.15%	75.03%	75.63%	77.45%	76.61%				

Figure 2: Comparing training strategies: noun chunk evaluation according to range and label

tive case instead of accusative case. The confusion between nominative and accusative case is due to the fact that both cases are expressed by identical morphology in the feminine and neutral genders in German. The morphologic similarity between accusative and dative is less substantial, but especially proper names and bare nouns are still subject to confusion. As the evaluation results show, the distinction between the cases could be learned in general, but morphological similarity and in addition the relatively free word order in German impose high demands on the necessary probability model.

5 Summary

We presented a German noun chunker for unrestricted text. The chunker is based on a head-lexicalised probabilistic context-free grammar and trained on unlabelled data. The base grammar was semi-automatically augmented with robustness rules in order to cover unrestricted input. An algorithm for chunk extraction was developed which maximises the probability of the chunk sets rather than the probability of single parses like the Viterbi algorithm.

German noun chunks were detected with 93% precision and 92% recall. Asking the chunker to additionally identify the syntactic category and the case of the chunks resulted in recall of 83% and precision of 84%. A comparison of different training strategies showed that unlexicalised parsing information was sufficient for noun chunk extraction with and without case information. The base grammar played an important role in the chunker development: (i) building the chunker on the basis of an already trained grammar improved the chunker rules, and (ii) refining the base grammar with even simple verb-first and verb-second rules improved accuracy, so it should be worthwhile to further extend the grammar rules. Increasing the amount of training data also improved noun chunk recognition, especially case disambiguation. Better heuristics for guessing the parts-of-speech of unknown words should further improve the noun chunk recognition, since many errors were caused by unknown words.

References

Steven Abney. 1991. Parsing by Chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle-Based Parsing*. Kluwer Academic Publishers.

Steven Abney. 1996. Partial Parsing via Finite-State Cascades. In *Proceedings of the ESSLLI '96 Robust Parsing Workshop*.

Franz Beil, Glenn Carroll, Detlef Prescher, Stefan Riezler, and Mats Rooth. 1999. Inside-Outside Estimation of a Lexicalized PCFG for German. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 269–276.

	Strategy 4					
	Initial		A		B	
	prec	rec	prec	rec	prec	rec
untrained	90.22%	90.18%	90.43%	90.60%	90.43%	90.60%
unlex1	92.84%	**91.58%**	91.65%	91.35%	91.52%	91.35%
unlex2	93.01%	91.49%	92.45%	**91.58%**	91.89%	91.67%
unlex3	92.95%	91.25%	92.64%	91.21%	92.21%	91.86%
unlex4	93.09%	90.79%	93.11%	91.07%	92.73%	91.86%
unlex5	**93.29%**	90.32%	**93.20%**	91.02%	92.73%	91.86%
unlex6					**92.91%**	91.96%
unlex7					92.83%	**92.10%**

Figure 3: Grammar and training data extensions: noun chunk evaluation according to range only

	Strategy 4					
	Initial		A		B	
	prec	rec	prec	rec	prec	rec
untrained	72.02%	71.98%	74.42%	74.56%	74.42%	74.56%
unlex1	75.87%	74.84%	78.51%	78.25%	77.60%	77.46%
unlex2	78.27%	76.99%	80.74%	**79.98%**	79.89%	79.70%
unlex3	78.70%	**77.27%**	81.24%	**79.98%**	81.17%	80.87%
unlex4	78.80%	76.85%	81.83%	80.03%	82.44%	81.67%
unlex5	**79.28%**	76.75%	**81.85%**	79.93%	82.53%	81.76%
unlex6					82.94%	82.09%
unlex7					**83.88%**	**83.21%**

Figure 4: Grammar and training data extensions: noun chunk evaluation according to range and label

Thorsten Brants. 1999. Cascaded markov models. In *Proceedings of EACL'99*.

Glenn Carroll and Mats Rooth. 1998. Valence Induction with a Head-Lexicalized PCFG. In *Proceedings of Third Conference on Empirical Methods in Natural Language Processing*.

Eugene Charniak. 1997. Statistical Parsing with a Context-Free Grammar and Word Statistics. In *Proceedings of the 14th National Conference on Artificial Intelligence*.

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting Syntactic Structure for Language Modeling. In *Proceedings of the 36th Annual Meeting of the ACL*.

Kenneth W. Church. 1988. A Stochastic Parts Program and Noun Phrase Parser for unrestricted Text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 136–143.

Michael Collins. 1997. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the ACL*.

Jason Eisner and Giorgio Satta. 1999. Efficient Parsing for Bilexical Context-Free Grammars and Head Automaton Grammars. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 457–464.

Joshua Goodman. 1996. Parsing Algorithms and Metrics. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 177–183.

Joshua Goodman. 1997. Probabilistic Feature Grammars. In *Proceedings of the 5th International Workshop on Parsing Technologies*, pages 89–100.

K. Lari and S. Young. 1990. The Estimation of Stochastic Context-Free Grammars using the Inside-Outside Algorithm. *Computation Speech and Language Processing*, 4:35–56.

Hermann Ney, U. Essen, and R. Kneser. 1994. On Structuring Probabilistic Dependencies in Stochastic Language Modelling. *Computer Speech and Language*, 8:1–38.

L. Ramshaw and M. Marcus. 1995. Text Chunking using Transformation-Based Learning. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94.

Mats Rooth. 1992. Statistical NP Tagging. Unpublished manuscript.

Anne Schiller and Chris Stöckert, 1995. *DMOR*. Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Helmut Schmid. 1999. Lopar: Design and Implementation. Technical report, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Pasi Tapanainen and Timo Järvinen. 1998. Dependency Concordances. *International Journal of Lexicography*, 11(3):187–203.

Atro Voutilainen. 1993. NPtool, a Detector of English Noun Phrases. In *Proceedings of the Workshop on Very Large Corpora*, pages 48–57.