Off-line (and On-line) Text Analysis for Computational Lexicography

Von der Philosophisch-Historischen Fakultät der Universität Stuttgart zur Erlangung der Würde eines Doktors der Philosophie (Dr. phil.) genehmigte Abhandlung

Vorgelegt von
Hannah Kermes
aus Esslingen

Hauptberichter: Prof. Dr. Christian Rohrer
Mitberichter: Dr. habil. Ulrich Heid


Institut für maschinelle Sprachverarbeitung
Universität Stuttgart

2003
Acknowledgements

The research reported on in this thesis was based on work in two projects: the DFG-project *Deutsches Referenzkorpus*, a joint project of the Institut für Maschinelle Sprachverarbeitung (IMS) in Stuttgart, the Seminar für Sprachwissenschaft (SfS) in Tübingen, and the Institut für deutsche Sprache (IDS) in Mannheim, and the DFG-Transferbereich project *Automatische Exzerption* which is intended to support the transfer of know-how from universities to companies (project partners are the Institut für Maschinelle Sprachverarbeitung (IMS) in Stuttgart, the Dudenredaktion in Mannheim and the Langenscheidt KG in München).

I would like to thank all the members of the IMS for providing an excellent and friendly working environment. In particular I wish to thank my supervisors Christian Rohrer and Ulrich Heid for fruitful discussions, for their feedback and motivation, and for the fact that they were always willing to lend me an ear.

I am grateful to Esther König, who was responsible for the DEREKO-project, for her permanent unofficial supervision. She encouraged me to go my own way without losing sight of the main issues. She was of great help discussing conceptual as well as technical issues, thus putting my research on a solid foundation.

Special thanks go to Stefan Evert, my office-mate, who not only discussed work matters with me, but is also responsible for implementing, and modifying tools I used for my research. I could not have realized many of the ideas I had, had he not implemented new features, and thus opened up new possibilities. In addition, I want to thank Wolfgang Lezius for providing an import filter for TIGERSearch for my chunker. I also want to thank Stefanie Dipper, Sabine Schulte im Walde, Heike Zinsmeister, and Arne Fitschen for giving me insights into their work, Kristina Spranger for writing a Dutch version of the chunker and for giving remarks on the grammar, and Piklu Gupta for revising my English.

Finally, I wish to thank my family and friends for the support, encouragement and distraction they provided me. And, last but not least, I want to thank Thomas for his support and for letting me forget about work.
Contents

1 Introduction
   1.1 Motivation: Extraction of lexicographic information ............ 1
      1.1.1 Information needed for dictionary upgrading ........... 3
      1.1.2 A corpus linguistic approach to lexicography .......... 5
   1.2 Approaches to text analysis .................................. 7
      1.2.1 Symbolic approaches ...................................... 8
      1.2.2 Probabilistic approaches ................................ 10
      1.2.3 Hand-written rules ....................................... 11
      1.2.4 Learning grammar rules .................................. 12
      1.2.5 On-line queries without previous parsing ............... 13
      1.2.6 Full Parsing ............................................. 15
      1.2.7 Chunking ................................................. 16
   1.3 Requirements ................................................... 17
      1.3.1 Requirements for the extraction tools .................... 17
      1.3.2 Requirements for the corpus annotation .................. 18
      1.3.3 Criteria for selection .................................... 19
   1.4 Classical chunking .............................................. 20
      1.4.1 State-of-the-art systems .................................. 20
      1.4.2 Problems for extraction ................................... 23
   1.5 Conclusion ....................................................... 26

2 General Concept of Off-line Analysis .............................. 29
   2.1 Introduction ..................................................... 29
   2.2 A recursive chunker for unrestricted German text ............. 29
   2.3 Technical framework .......................................... 33
      2.3.1 The IMS Corpus Workbench (CWB) ......................... 33
      2.3.2 The Corpus Query Processor (CQP) ....................... 33
      2.3.3 Applying and processing the rules ....................... 38
      2.3.4 Output formats .......................................... 42
      2.3.5 Advantages of the architecture ........................... 43
   2.4 General framework of YAC .................................... 45
      2.4.1 Linguistic coverage ..................................... 45
      2.4.2 Feature annotation ..................................... 50
# Contents

2.4.3 Chunking process ........................................ 52

3 Grammar Details .............................................. 63
  3.1 Introduction .................................................. 63
  3.2 Simple phrasal categories .................................. 67
    3.2.1 Adverbial Phrases (AdvP) .............................. 67
    3.2.2 Verbal Complexes (VC) ................................. 70
  3.3 Complex phrasal categories ................................ 73
    3.3.1 Adjectival Phrases (AP) ............................... 73
    3.3.2 Noun Phrases (NP) ..................................... 91
    3.3.3 Prepositional Phrases (PP) ............................ 121
  3.4 Subordinate clauses ....................................... 132

4 Evaluation ..................................................... 141
  4.1 Introduction ................................................ 141
  4.2 Evaluating a chunker ...................................... 142
  4.3 How to get a real gold standard .......................... 145
    4.3.1 A German treebank: NEGRA .......................... 145
    4.3.2 A manually annotated corpus .......................... 157
    4.3.3 Discussion .............................................. 161
  4.4 Comparing different chunkers and partial parsers .......... 161

5 Applications of YAC-chunked Corpora ......................... 165
  5.1 Introduction ............................................... 165
  5.2 Sample extractions ........................................ 165
    5.2.1 Subcategorization properties of adjectives ........... 166
    5.2.2 Collocation information .............................. 177
    5.2.3 Extraction of named entities .......................... 181

6 Conclusion .................................................... 187

Zusammenfassung ................................................ 191
  6.1 Voraussetzungen ........................................... 191
  6.2 YAC - Ein rekursiver Chunker für unbegrenzten Deutschen Text . 192
    6.2.1 Annotierte Strukturen ............................... 192
    6.2.2 Feature Annotation ................................... 195
    6.2.3 Chunkingprozess ..................................... 196
  6.3 Evaluierung ............................................... 197
    6.3.1 Evaluierungsergebnisse ................................ 198
  6.4 Anwendungen .............................................. 199
## List of Tables

2.1 Feature annotation for all chunk categories ........................................ 50

3.1 Annotation scheme of ADVP (adverbial phrase) ................................. 67
3.2 Lexical-semantic properties of ADVP (adverbial phrase) .................... 67
3.3 First level ADVP rules ........................................................................ 68
3.4 Third level ADVP rules ....................................................................... 69
3.5 Annotation scheme of VC (verbal complexes) ....................................... 70
3.6 Lexical-semantic properties of VC (verbal complexes) .......................... 71
3.7 First level VC rules ............................................................................ 72
3.8 Annotation scheme of V (verb) .............................................................. 72
3.9 Lexical-semantic properties of V (verb) ................................................ 73
3.10 First level V rules .............................................................................. 73
3.11 Annotation scheme of AP (adjective phrase) .......................................... 75
3.12 Lexical-semantic properties of AC (adjectival chunk) ......................... 76
3.13 Lexical-semantic properties of AP (adjective phrase) ......................... 77
3.14 First level AP rules ............................................................................ 78
3.15 Second level AP rules ......................................................................... 81
3.16 Third level AP rules ........................................................................... 91
3.17 Annotation scheme of NP (noun phrase) .............................................. 92
3.18 Lexical-semantic properties of NC (noun chunk) ................................. 95
3.19 Lexical-semantic properties of NP (noun phrase) ................................ 97
3.20 First level NC rules ........................................................................... 98
3.21 First level NP rules ........................................................................... 108
3.22 First level NPRON rules .................................................................... 109
3.23 Second level NCC rules ................................................................... 111
3.24 Second level NP rules ....................................................................... 115
3.25 Third level NP rules .......................................................................... 119
3.26 Annotation scheme of PP (prepositional phrase) ................................. 122
3.27 Lexical-semantic properties of PP (prepositional phrase) .................. 124
3.28 First level PP rules ........................................................................... 125
3.29 First level PPPRON rules .................................................................. 128
3.30 Second level PP rules ....................................................................... 129
3.31 Third level PP rules .......................................................................... 132
3.32 Annotation scheme of CL (clause) ...................................................... 134
List of Tables

3.33 Lexical-semantic properties of CL (clause) .................................. 134
3.34 Clause level CL rules ................................................................. 135

4.1 Evaluation figures of YAC on ideal PoS-tags .............................. 158
4.2 Evaluation figures of YAC on automatic PoS-tags ....................... 159
4.3 Evaluation figures of the best version of Schiehlen’s full noun
   chunker ......................................................................................... 162

5.1 adjective+verb+finite clause ....................................................... 175
5.2 adjective+verb+finite clause sorted by phenomena ...................... 177
5.3 Results for Adj+N extraction task ............................................. 178

6.1 Feature Annotation der einzelnen Chunkkategorien .................... 195
6.2 Evaluierungswerte von YAC auf idealen Tags ......................... 198
6.3 Evaluierungswerte von YAC auf automatisch annotierten Tags .. 199
6.4 Feature annotation for all chunk categories .............................. 206
6.5 Evaluation figures of YAC on ideal PoS-tags .............................. 209
6.6 Evaluation figures of YAC on automatic PoS-tags ...................... 209
# List of Figures

1.1 On-line vs. off-line approaches ........................................... 6

2.1 TIGERSearch tree diagram of (2.1a) ................................. 31
2.2 Interaction among CQP and Perl-scripts ................................. 38
2.3 Morpho-syntactic information of the elements of the NP *den vierten Tag* ....................................................... 41
2.4 Example sentence in the XML-output-format .............................. 44
2.5 TIGERSearch tree diagram of (2.11) .................................. 47
2.6 Parsing Architecture of YAC ............................................. 53
2.7 TIGERSearch tree diagram of (2.24b) ................................ 56
2.8 TIGERSearch tree diagram of (2.25b) ................................ 57
2.9 TIGERSearch tree diagram of (2.29) ................................ 60

3.1 Cycle of embedding ...................................................... 66
3.2 TIGERSearch tree diagram of (3.13c) ................................. 74
3.3 TIGERSearch tree diagram of (3.21) ................................ 77
3.4 TIGERSearch tree diagram of (3.22) ................................ 78
3.5 TIGERSearch tree diagram of (3.40) ................................ 85
3.6 TIGERSearch tree diagram of (3.41) ................................ 86
3.7 TIGERSearch tree diagram of (3.42) ................................ 88
3.8 TIGERSearch tree diagram of (3.43) ................................ 88
3.9 TIGERSearch tree diagram of (3.47d) ................................ 93
3.10 TIGERSearch tree diagram of (3.47e) ................................ 94
3.11 TIGERSearch tree diagram of (3.90) ................................ 115
3.12 TIGERSearch tree diagram of (3.91) ................................ 117
3.13 TIGERSearch tree diagram of (3.95c) ................................ 123
3.14 TIGERSearch tree diagram of (3.95c) ................................ 124
3.15 TIGERSearch tree diagram of (3.110) ................................ 130
3.16 TIGERSearch diagram of (3.112a) ................................ 133

4.1 NEGRA analysis of (4.1a) ............................................. 145
4.2 NEGRA analysis of (4.1b) ............................................. 145
4.3 NEGRA analysis of (4.2) ............................................. 147
4.4 NEGRA analysis of (4.4) ............................................. 149
List of Figures

4.5 NEGRA analysis of (4.4b) ........................................ 149
4.6 YAC analysis of (4.6) ........................................ 151
4.7 NEGRA analysis of (4.10) ..................................... 154
4.8 YAC analysis (4.10) ........................................ 155

5.1 Extraction results for adjective + PP for zuständig .......... 169
5.2 Extraction results of adjective+verb+clause for klar ..... 176

6.1 Architektur des Chunkprozesses ............................. 197
6.2 Chunking architecture ........................................ 207
Chapter 1

Introduction

1.1 Motivation: Extraction of lexicographic information

There is a rising interest in use of parsing tools and the resulting corpus annotation in computational lexicography. However, the annotation delivered by existing parsing tools does not always meet the needs of lexicographic and linguistic research. The question is what the requirements for such a tool and the resulting annotation are. Let us have a look at three different kinds of dictionaries:

- dictionaries for human use
  - printed monolingual dictionaries
  - electronic dictionaries
- machine readable dictionaries for Natural Language Processing (NLP) systems

Printed monolingual dictionaries  Printed monolingual dictionaries intend to cover most important semantic and syntactic aspects of words. Maintenance of consistency and completeness of these dictionaries is a problem for the following reasons:

- information is missing, i.e., important information (e.g. collocations) is not available
- entries are incomplete, i.e., not all relevant information for a certain entry is listed
- information is not consistent, i.e., relevant information is listed for one entry but not for another
Introduction

- language changes make it necessary to modify or extend entries, i.e., words have changed with respect to their syntactic behavior, their meaning, and collocational usage.

Electronic dictionaries for human use  Electronic dictionaries as new media offer the possibility to make more, more detailed, and different kinds of information available to the user without losing transparency:

- enormous amounts of information can be stored in a compact format
- search engines allow for easy and fast access to the desired data
- users can choose how much and what kind of information they are interested in
- a reference corpus can be made available as a different kind of dictionary

In paper dictionaries information is left out in order to save space. Electronic dictionaries have the possibility to store large amounts of data in a compact format. Search engines provide an easy and fast access to the desired data. Thus, there is no need to neglect information.

Another reason to reduce the amount of information in paper dictionaries is transparency. If too much information is put into a lexical entry, it is difficult for the user to find the desired information. Electronic dictionaries, however, can offer the possibility of user profiles. In other words, the user can decide how much and what kind of information he is interested in. Whether he wants to see subcategorization information, selectional preferences, synonyms or morphosyntactic information. A reference corpus additionally offers the possibility to see the usage of a word in real language data. However, the issues of consistency and completeness still apply in principle.

Machine readable dictionaries for Natural Language Processing (NLP)  Natural Language Processing (NLP) applications such as full parsers and machine translation (MT) systems need detailed and consistent information about words. In order to be able to parse sentences correctly or to determine the correct translation, the structure of a sentence has to be analyzed with all its local and long distance relations, and possible ambiguities have to be resolved. Detailed knowledge about subcategorization frames, selectional preferences, collocations, idiomatic usage, and specific syntactic construction types is necessary. If we take adjectives in predicative(-like) constructions as an example$^1$, parsing systems need the following detailed information:

- adjectives occuring in predicative(-like) constructions

$^1$We will refer to this example more often during this book.
1.1 Motivation: Extraction of lexicographic information

- adjective + verb cooccurrences
- selectional preferences verb + adjective
- predicative-like constructions with non-copula verbs

1.1.1 Information needed for dictionary upgrading

We need detailed information about words to improve dictionaries. The information includes:

1. syntactic information
   - subcategorization patterns such as predicative adjectives and copula verbs subcategorizing subordinate clauses as in (1.1)
   - syntactic construction types such as:
     - predicative-like constructions with adjectives and non-copula verbs as in (1.2a)
     - elliptical constructions of predicative adjectives as in (1.2b+c)
   - grammatical functions:
     - predicative usage of adjectives

2. semantic information
   - selectional preferences such as:
     - preference of certain predicative adjectives for certain copula verbs
   - collocations such as:
     - noun + verb
     - adjective + noun
     (rote Rosen (red roses))
     - noun + preposition
   - synonyms
   - multi-word units
   - lexical classes, e.g., temporal nouns, directional adverbs

3. morphological information
   - morpho-syntactic information (e.g., case, number, gender, definiteness)
   - compounding and derivation
Introduction

Syntactic information

Syntactic information about words is important for dictionaries for human usage as well as for lexicons for NLP applications. Subcategorization information, e.g., about predicative adjectives and copula verbs subcategorizing subordinate clauses is necessary to determine the correct analysis of sentences as in 1.1, which show the predicative adjective *deutlich* (clear) with different copula verbs.

(1.1) a. [...] machte er deutlich, daß die Wiederverwertung nur zu Produkten minderer Qualität führt. ' [...] he made clear, that recycling will only lead to products of inferior quality.'

b. [...] werde es deutlich, daß Deutschland ein Antidiskriminierungsgesetz brauche. ' [...] it becomes clear, that Germany needs an anti-discrimination law'

c. Es ist deutlich, daß wir uns nicht verstehen. 'It is clear, that we us not understand.'

It is equally important to know about certain syntactic construction types. For example about predicative-like constructions as in (1.2a), where the adjective *deutlich* appears together with the non-copula verbs *zeigen* in a predicative-like construction with the reflexive pronoun *sich* (oneself). Other predicative adjectives such as *gut* and *klar* can occur in elliptical constructions without the copula verb as the examples in (1.2b–c) show.

(1.2) a. Es zeigt sich deutlich, daß wir uns nicht verstehen. 'It shows itself clear, that we us not understand.'

b. (Gut) möglich, daß er ihn gesehen hat. 'It is (well) possible, that he has seen him.'

c. (Schon) klar, daß ihm das gefällt. 'It is clear that he likes it.'
1.1 Motivation: Extraction of lexicographic information

Semantic information

Selectional preferences are also important information. In general predicative adjectives show a clear preference for occurring with the copula verb sein (to be). There are, however, predicative adjectives which show preferences for occurring with other copula verbs. The adjective deutlich (clear), e.g., is relatively often found with the copula verbs machen (to make) (1.1a) and werden (to become) (1.1b), while the construction with sein (to be) (1.1c) is marginal. In this case, we are interested in the semantic aspect, and not in the syntactic difference.

1.1.2 A corpus linguistic approach to lexicography

In order to meet the needs of the maintenance of consistency and completeness within a lexicon, lexicography is moving away from solely manually constructed dictionaries to computer-assisted methods. Lexical engineering aims towards a scalable lexicographic work process for large lexica. It ensures that the processes are reproducible on large amounts of data. Besides, quality standards that are valid for large lexica assure the quality of the product. Thus, automatic acquisition of linguistic and lexicographic knowledge can make lexicographic work easier, and faster, and it can help to maintain consistency and completeness within the lexicon (cf. Underwood 1998; Kilgarriff and Tugwell 2001a; Kilgarriff and Tugwell 2001b).

The information needed to enhance existing dictionaries is inherent in any language performance, e.g., in text corpora. In other words, text corpora can serve as a knowledge source, as a new kind of dictionary.

The problem is, how to make the text-inherent knowledge available, i.e., how can we extract relevant information from the text corpora. There are principally two possible approaches:

1. simple on-line analysis

2. a combination of off-line and on-line analysis

In the case of a mere on-line analysis, extraction queries are applied to a corpus which is not annotated or has annotations on token level only, i.e., the corpus is tokenized, lemmatized, part-of-speech-tagged and possibly enriched with morpho-syntactic information. All of the syntactic analysis is left to the extraction queries. In other words, the parsing is performed on-line. In the case of a combination of off-line and on-line analysis the corpus is annotated syntactically before extraction queries are applied. Depending on the depth of the off-line analysis, the extraction queries have to perform more or less of the parsing needed for extraction. The off-line analysis can consist of a partial syntactic analysis (chunking or partial parsing) or a full hierarchical analysis (full
Figure 1.1: On-line vs. off-line approaches
1.2 Approaches to text analysis

The different approaches to text analysis can be differentiated with respect to three different dimensions. These dimensions can be seen as three axes. In between these three axes lie the approaches.

1. type of grammar
   
   (a) symbolic grammars, where the rules use symbolic elements as trigger for the correct analysis
   
   (b) probabilistic grammars, where symbolic rules are enriched with probabilities, and statistical means are used to determine or select the correct analysis

Hypothesis The better and more detailed the (off-line) annotation, the better and faster the (on-line) extraction. However, the more detailed the (off-line) annotation, the more complex the grammar, the more time consuming and difficult the grammar development, and the slower the parsing process.

In other words, rich corpus annotation on token and on structural level provides detailed information. The extreme in this case is treebank with full hierarchical analysis, which is manually corrected. Extraction queries which are applied to a corpus with full hierarchical annotation can more or less collect the information provided by the off-line tool. The advantage is that the extraction queries can be kept relatively simple, and are easy to develop. The results of the queries can be expected to be reliable. The disadvantage is that a rich off-line analysis is costly. It requires a lot of manual work for grammar development, and possibly manual correction or disambiguation of the annotation.

A less sophisticated off-line analysis is less time consuming and labor intensive with respect to grammar development. The off-line analysis is faster, and more robust. However, more of the analysis is left to the queries, which makes them more complex. Consequently, more time and manual work is needed for the development of the queries. Besides, the results of the queries are less reliable and may require manual selection. The question, thus, is, what is better all in all. Does the manual work spared with respect to grammar development outweigh the increased manual work for query development and result checking, or is it vice versa?
2. type of grammar development

(a) hand-written grammars, where a grammar writer develops the rules by hand

(b) learning methods, where the grammar rules are learned using algorithms

3. depth of analysis

(a) complete on-line analysis with no pre-query annotation on syntactic level

(b) full parsing, i.e., a full hierarchical, relational, and functional analysis

(c) chunking, i.e., a partial syntactic analysis of variable depth

1.2.1 Symbolic approaches

A symbolic approach allows us to formulate precise rules. Symbolic elements are used as triggers to determine the correct analysis. Analysis can be easily reconstructed and eventual modifications or corrections can be made. Lexical knowledge can be excluded as additional triggers. The results of rules can be predicted and controlled.

In some cases, however, symbolic triggers are not sufficient to solve ambiguities. Long distance context information or world knowledge would be necessary for disambiguation. Long distant context information, however, is difficult to implement, and the necessary relation cannot be established with enough reliability. In these cases, symbolic approaches can only use heuristics to determine the correct analysis, accepting, at the same time, that there might be false decision. The other possibility is to leave the constructions ambiguous, and return all possible analyses. In some cases, this can result in a large parse forest.

Complex grammars make use of comprehensive grammatical and lexical knowledge to be able to deal with all language possibilities. Symbolic grammar rules allow formulation of detailed rules precisely. This makes the grammar elaborate but also complicated. Interactions among rules are often difficult to predict and to control. The extensive use of linguistic and lexicographic knowledge not only resolves but also produces ambiguities. Development and modification of the rules is costly, and difficult. Besides, the comprehensive rules and the complex rule system can affect the performance of the system, and can slow down the parsing process.

Symbolic grammars usually do not support underspecification. The single rules of the grammar are precisely formulated, and only those phenomena
which are explicit in the grammar rules can be dealt with. Thus, symbolic approaches often lack robustness.

Unification-based grammars

Unification-based grammars (e.g., in the LFG (Bresnan 2001; Bresnan 1982)\(^2\) or HPSG framework (Pollard and Sag 1987; Pollard and Sag 1994; Sag and Wasow 1999)) are usually complex grammars which are able to model the hierarchical structure of language, as well as to handle attachment ambiguities, determine relations among constituents, and to determine their functions. Therefore, they need a rich and complex rule system, where the grammatical aspects of a language are precisely formulated. The richness of the rule system, however, results in complex interactions among the rules, which have to be considered during grammar development and modification as well as for adjustments of the grammar to particular domains.

In order to deal with ambiguities and the various structural possibilities constraint-based grammars make use of extensive lexical information. Possible structural analyses have to conform to the information present in the lexical database. This can pose a problem, if the information needed is missing. In this case, the grammar cannot provide a valid analysis.

The richness and complexity of rules and information used, however, does not only resolve ambiguities, but produces them as well. Usually, a large number of possible analyses is provided for each sentence, which cannot be stored and queried efficiently for large corpora. Thus, an additional disambiguation component has to be applied. This can either be statistical or manual. The former does not necessarily provide the correct analysis, the latter is labor-intensive and costly to produce.

Context-free Grammars (CFG)

Context free Grammars (CFG) (see Langer 2001) are formal grammars consisting of a set of recursive rewriting rules. The rewriting rules (or productions) replace a terminal and non-terminal symbols by a terminal symbol. In contrast to some other symbolic approaches CFGs are in general relatively easy and fast to develop. The grammar is modular with only minimal interaction among the rules. The parsing process is usually fast, which makes it feasible to work with large amounts of text.

The grammar, however, can only be kept simple and fast, as long as it covers only the basic aspects of a language system. For the automatic analysis of large amounts of text, further so-called robustness rules are required to overcome

\(^{2}\)The PARGRAM grammar developed within the PARGRAM-Project at the IMS in cooperation with Xerox is an example of a large-scale LFG-grammar for German. An evaluation of the parser is given in Dipper 2000
shortcomings in the grammar. This, however, slows down the parsing process and makes the grammar unwieldy.

The output of CFG parsers does not include much information. In order to include, e.g., information about agreement or lexical properties, a large number of additional rules is required. The implementation of these rules within the CFG framework is difficult, especially, with respect to disambiguation of agreement features during the parsing process. The more aspects and features are to be considered, the more complicated and complex the rule system, and the more difficult the development of the grammar.

In other words, the CFG framework works well for small grammars which do not have to cover a wide spectrum of phenomena and which do not have to deliver a lot of additional information apart from the structural annotation. If a more complex grammar and/or additional information about the properties of constructions is required, the CFG framework loses its advantages.

1.2.2 Probabilistic approaches

Probabilistic grammars use statistical means to determine the correct analysis (see chapters 11 and 12 of Manning and Schütze (1999) for an overview). Hand-written or learned rules are trained on a corpus, i.e., the rules are applied to a corpus, frequency data is collected, and the probability for each rule is calculated. The grammar rules are then expanded adding the probabilistic information acquired from the training corpus to the rules. The training can be either supervised or unsupervised. Supervised training means that the training data is a manually corrected corpus. Unsupervised training is performed on corpus data which is unannotated.

During the parsing process, all possible analyses are produced, and the probability of each analysis is calculated. If there is more than one analysis, statistical means are used to determine the most probable or the $n$ most probable analysis.

The use of statistical methods makes it possible to keep the rules relatively simple. There is no need for comprehensive lexical and linguistic knowledge that symbolic approaches demand. Instead of detailed and complex rules, probabilities can be used to determine the correct analysis. The rules can even be left underspecified. Underspecification of rules allows us to deal with phenomena, which are not overtly and actively present in the grammar. This makes probabilistic grammars robust, as they can cover phenomena which are not explicitly dealt with in the grammar.

Probabilistic grammars depend on the training corpus they use. Probabilities can only be calculated according to the representation of the different phenomena in the corpus. If the training corpus is not well balanced, the probabilities for the single rules can be shifted, i.e., they do not reflect normal language use
but frequency distributions. But even if the training corpus is well balanced, some phenomena are better represented than others. High frequency phenomena are well represented, while low frequency phenomena are less well represented. If highly frequent and low frequency phenomena compete with each other, the highly frequent phenomena will in general be preferred over the low frequency phenomena. This is also the case, if a phenomenon has lower frequency figures in comparison to another phenomenon.

An example: Probabilistic context-free grammars (PCFG)

A probabilistic context-free grammar (PCFG) (see Chapter 11 of Manning and Schütze (1999) for an overview) basically has the same rule system as a CFG. The only difference is that each rule is enriched by information about the probability of the rule. In order to acquire the probabilities for each rule, the rule system is applied to a training corpus. PCFGs have the advantage that they can use underspecification in their rules. However, they are not as fast as CFGs as they involve more information. This is even more the case for Probabilistic Head-lexicalized context free grammars. An example of a head-lexicalized context free grammar for German is the GRAMOTRON grammar (Schulte im Walde et al. 2001). Within the framework of GRAMOTRON, all grammar rules are indexed by the lemma of the respective syntactic head. The rule system in the beginning is comparatively simple. It consists of a number of hand-written rules, which during training are multiplied by the occurring lexical heads. This leads to a high number of parameters, which need a considerable amount of working memory a fact that slows down the parsing speed considerably. The GRAMOTRON grammar is unsupervised, i.e., it does not need an annotated corpus to train. Consequently, it is not so much dependent on the training corpus as the grammar can train anew for each corpus that is to be parsed.

Usually, the extraction in the case of GRAMOTRON does not take place on the corpus itself but on the trained, and lexically enriched grammar rules. In other words, once it is trained on a corpus, the grammar itself is a kind of lexicon. There is, however, the possibility of performing the extraction on Viterbi parses (Zinsmeister and Heid 2003).

1.2.3 Hand-written rules

Hand-written grammars are grammars where the rules are developed by a grammar writer. He decides what kind of rules the grammar comprises, and which phenomena the grammar covers. Consequently, the grammar writer has a good control of the rule system, i.e., what the output of the grammar looks like, what kind of language data it can processed, and what not.

At the same time, the grammar depends heavily on the expertise of the grammar writer. Only those phenomena are represented in the grammar which
the grammar writer has thought of. This is especially the case, if the grammar is based on symbolic triggers, as symbolic grammars usually do not allow underspecification.

Hand-written rules can be used for symbolic grammars, and as a basis for probabilistic grammars. In the latter case, the rules are applied to a corpus after they have been developed by the grammar writer in order to calculate the probabilities for each rule. In the case of head lexicalized probabilistic grammars the rules are additionally multiplied by the occurring lexical heads.

1.2.4 Learning grammar rules

An alternative to writing grammar rules by hand are machine learning techniques. Machine learning algorithms try to infer grammar rules from text corpora. Extensional syntactic descriptions (corpus annotation) are turned into intensional descriptions (rules). Approaches based on learning can either work on optimal or on suboptimal training data. In the former case, the algorithms try to infer the grammar based on correctly analyzed text (usually treebanks). In the latter case, the grammar rules are built on the basis of annotations which are the result of an automatic process and which can include incorrect analysis.

The big advantage of machine learning is that new resources in the form of text corpora can be exploited to detect a grammar, i.e., the manual work goes into the corpus annotation instead of the rule writing. The grammar resulting from such machine learning techniques can either be symbolic or probabilistic. The grammar a machine learning algorithm generates is more or less independent of the grammatical knowledge of the grammar developer as it is detected from the training data rather than being a result of the expertise of the human developer. This, however, makes the grammar depend on the corpus it is derived from, or better on the constructions and linguistic information this corpus includes. Consequently, the grammar covers only those phenomena present in the training data.  

A special case: memory based learning

Memory based learning algorithms can be seen as a special case of learning. Most prominent is the data oriented parsing approach (DOP). The DOP approach extracts tree fragments (so-called subtrees) from structurally annotated text corpora. The fragments are stored and as such replace the grammar. Language generation and language analysis is performed by combining the mem-

---

3Interesting research comparing different learning methods, their benefits, advantages and disadvantages was conducted by a post-Doc research group in the Netherlands (Learning Computational Grammars (TMR-Network))

4See (Bod and Scha 1997) for an overview.
orized fragments building up a complex hierarchical structure. In the case of parsing, the final structure has to fit the input data.

DOP approaches need a structurally annotated corpus to collect the tree fragments that make up the grammar. Similar to other learning techniques, the grammar covers only those phenomena that can be dealt with using the fragments present in the training corpus. The evaluation figures that Rens Bod reports for his DOP model taken from different corpora indicate that the training corpus has great impact on the performance of the system. He reports precisions ranging from 83% to 94%\(^5\).

A solution seems to be to use a larger training corpus, which presumably will include more types of subtrees. However, memory based learning techniques are more sensitive to suboptimal data, i.e., they perform much better if the training corpus has correct structural analyses. Corpora with correct, manually checked, structural annotation are costly to develop and exist only in limited amounts\(^6\). Besides, memory based approaches make use of a large storage to memorize all the collected data. The more data has to be stored, the larger the storage capacity has to be. Consequently, memory based approaches are limited by and depend on storage capacity.

### 1.2.5 On-line queries without previous parsing

An approach where on-line queries are performed on a corpus without a previous annotation of structural attributes can be seen as an on-line analysis. The actual analysis has to be performed by the extraction query itself. On-line queries are usually a form of pattern matching in the sense of Kwic. Sequences of words, lemmas or PoS-tags are searched for. Depending on the desired result, these patterns are more or less detailed and complex. As the queries themselves are responsible for the analysis of the text, this approach is completely flexible. It does not depend on a previous structural annotation and its underlying theoretical background. Consequently, on-line queries are more or less theory independent with respect to syntactic theory. In principle, for each extraction a new theoretical background can be taken into account as each query can be adapted individually to meet the need for a specific application. Besides, the queries can be easily adapted to different text types. The approach works on basically any corpus without time consuming annotation. Some but not all on-line extraction systems require simple annotation on token level, such as PoS-tags and lemma information.

\(^5\)The bias and consistency of the DOP estimation model has been recently questioned by Mark Johnson (Johnson 2002).

\(^6\)Brants reports (Brants 2000) about 10 minutes/sentence for manual annotation (annotated twice and cross-checked) based on a graphical interaction of human annotators with a tagger and a parser running in the background.
On the other hand, this flexibility poses problems. For each extraction query, a complete analysis has to be performed. In other words, a repetition of an extraction query leads to a repetition of all of the sub-analyses. General elements, such as phrase structures, that occur in more than one query have to be analyzed anew for each query. This is especially undesirable, if a query includes complex structures, and if several queries are to be applied to the same corpus. Queries searching for complex constructions are usually complex themselves. If such a query has to perform all of the analysis, it is difficult to develop and modify as it gets more and more obscure. Precise restrictions on the analysis are difficult and costly to formulate, if possible at all. Consequently, in order to keep the query transparent for the user the single elements constituting the query are often restricted with respect to their complexity. This often means that the phrasal structures included are restricted to simple constructions. The simple structures also help to achieve a good precision for the extraction results, as only the simpler and therefore more secure constructions are taken into account. However, as many constructions are not considered, it affects the recall, which is then rather low. Yet, even if the single elements are kept simple, the query itself can become very complex, if it searches for a complex phenomenon.

An example: The work of Judith Eckle-Kohler 1999

An example of such a pattern matching strategy is the work of Judith Eckle-Kohler (Eckle-Kohler 1999). She developed a system for the automatic extraction of subcategorization information for German verbs from text corpora. As input she uses corpora which are pre-processed: tokenized, lemmatized and part-of-speech tagged. On this input she performs pattern matching queries built from regular expressions. In order to get a modular approach to the notion of adjuncts and complements, she models the general structures of a sentence as a sequence of open positions in which realizations of complements and adjuncts can occur. The model she uses follows the topological field model (Höhle 1986). Consequently, she has to describe the building blocks only once in her grammar. Nevertheless, these rule blocks have to be processed for each instance. A rule block appearing several times in a query has to be executed several times. The same holds for rule blocks that occur in several queries.

The single building blocks are kept relatively simple and restricted in order to achieve high quality results (Eckle reports an average precision of 93.67% for the extraction of transitive verbs in verb second clauses). Simple in this sense means that the building blocks are restricted with respect to the elements they can include. The elements of building blocks have to be clearly identifiable, constructions which can be ambiguous are left aside. Some complex constructions inhibit a unambiguous identification, consequently, they cannot be included. Only if the actual representation of a phenomenon consists of
these restricted building blocks can the extraction queries be successful, i.e.,
return results. Eckle allows, e.g., only definite genitive NPs to act as post-head
genitive modifiers, if they do not occur in topicalized position. This excludes
NPs as in (1.3).

(1.3) die Meinung der Gewerkschaft der Polizei (GdP)
     the opinion of the union of the police (GdP)

Most restrictive, however, is that Eckle consideres only those sentences,
where all NPs can be reliably categorized with respect to their case. If the
case of an NP is ambiguous, she ignores the NP. Consequently, she achieves a
low recall (around 2%).

In order to get a considerable amount of results, she has to work with very
large corpora. However, even as the single building blocks are kept simple, her
queries are usually rather complex. Besides, the analysis of the building blocks
has to be performed for each instance. Consequently, executing a query on a
very large corpus is time consuming.

1.2.6 Full Parsing

Another possible approach is to analyze the corpus with a full parser. A full
parser is based on a complex grammar that can be formulated in various frame-
works (e.g., Lexical Functional Grammars (LFG) or Head Driven Phrase Struc-
ture Grammars (HPSG)). These complex grammars are able to model hierar-
chical structures of a language. They are powerful enough to handle complex
constraints about structures, relations, and attachments. Consequently, they
are well suited to handle the problem of attachment ambiguities. In general,
they make use of detailed knowledge about the function and usage of words
to determine the correct analysis of a sentence. As a result a complete hier-
archical annotation is delivered providing rich and complex information about
structures, relations and functions.

The rich and complex annotation provides an excellent base for extraction
of linguistic and lexicographic information. All the knowledge inherent in a full
syntactic analysis is already available and only has to be collected by extraction
queries. Queries performed on such a basis can be kept simple and nevertheless
provide good results with respect to precision and recall.

The complexity of the grammars and the use of detailed linguistic and lexi-
cographic knowledge during the parse, however, slow down the parsing speed.
Thus, parsing large corpora is time consuming. Besides, if necessary informa-
tion is lacking, full parsers fail to deliver an analysis. In most cases, even if only
part of a sentence cannot be analyzed, the whole sentence is neglected. The
lack of robustness is often caused by a lack of linguistic or lexicographic knowl-
edge in the lexicon used by the parser. Full parsers depend heavily on a rich
Introduction

and detailed prerequisite lexicon which provides them with the necessary information. This is a problem for lexicographic applications, as the information that is to be extracted is needed for the parsing already.

The number of grammar rules produces ambiguities. In other words, the parser provides several possible analysis of which only one is correct. Some parsers use heuristic or statistical methods to determine the correct parse or to reduce the number of possible parses. Statistical parsers can provide probabilities for each of the possible parses. Extraction queries have to operate on the ambiguous output or on the viterbi parse provided by probabilistic parsers. The extraction queries collect all possible instances matching the query or the n-best instance. In the case of HPCFGs, the extraction is not performed on the corpus but rather on the grammar itself, which incorporates all the necessary information.

Another possibility to overcome ambiguous or incorrect output, is manual correction. As manual correction is time consuming and costly it is almost only used to build a reference corpus. This corpus, which is usually a treebank 7, is used to train, test and evaluate other NLP applications. The limited size of the treebank makes it insufficient for large scale extractions.

The complex rule system of the underlying grammar can also cause problems. Changes within the grammar can result in unexpected and undesirable interactions among rules. It is usually difficult to determine the cause of the interaction because of the complexity of the rules and the rule system. Consequently, it is very complicated and time consuming to modify and adjust the existing grammar to a new text domain or purpose.

1.2.7 Chunking

Another approach to text analysis which has become more and more popular is chunking or chunk parsing. The structures annotated by chunkers are usually non-hierarchical and non-recursive. As the annotated structures are relatively simple, the grammar itself is also relatively simple. In general, it does not consider cases resulting in ambiguities such as, e.g., attachment decisions, or those which include lexical dependencies. Rich and complex linguistic and lexicographic information is not required. As the name suggests, chunkers do not aim at annotating the structure of a sentence completely, but try to build

---

7Treebanks exist for a variety of languages, such as the UPenn Treebank for English (UPennTreebank ; Marcus et al. 1993; Marcus et al. 1994), TALANA for French (Abeillé et al. 2000; Abeillé et al. 2001), the Prague Dependency Treebank (Prague Dependency Treebank ; Hajicova and Sgall 2001), the Bultreebank for Bulgarian (BultreeBank ; Simov et al. 2002; Simov et al. 2001) and NEGRA (NEGRA Project ; Skut et al. 1997; Skut et al. 1997; Skut et al. 1998) , TIGER (TIGER ; Brants et al. 2002; Brants and Hansen 2002; Lezius 2001) and Verbmobil (Stegmann et al. 1998) for German.
“chunks” of words. Consequently, the rule system of chunkers is relatively simple, and they are very robust, i.e., they are not apt to fail to parse a sentence because they fail to parse part of the sentence. A chunker analyzes a text as far as it can, and annotates the results.

The output of a chunker can be seen as somewhat in between a corpus with standard preparation (tokenizing, lemmatizing, PoS-tagging) and a full parse. It provides an annotation on which an extraction query can be built and which takes more or less of the analysis from the query, depending on the underlying chunk definition.

### 1.3 Requirements

#### 1.3.1 Requirements for the extraction tools

The requirements for a system which aims to provide a useful basis for extractions are the following:

1. the system has to work on unrestricted text
   - no limitation to corpus size, i.e., it should be able to deal with small as well as large corpora
   - it should be able to parse complete sentences as well as fragmentary text (e.g., NPs)
   - the system should not be domain specific, i.e., it should work basically on any text type
   - additional domain specific rules should be easy to incorporate in the grammar

2. shortcomings in the grammar should not lead to a complete failure to parse

3. no manual checking should be required as this is not feasible for large quantities of text

4. the system should provide clearly defined and documented interfaces for extraction processes:
   - the annotation should make use of linguistic standards
   - there should be a documentation on what is annotated, and how it is annotated
   - the annotation should be easy to process for further application

---

8For a more detailed definition of chunk see 1.4.
1.3.2 Requirements for the corpus annotation

In order to provide a useful basis for extraction processes, the corpus annotation should include, besides the information on token level:

1. the head lemma of annotated structures to determine the lemma of the future dictionary entry,
2. morpho-syntactic information, to determine the grammatical function of the structure and to extract additional aspects of potential lexical entries, e.g., singular plural alternations for nouns,
3. lexical-semantic information, e.g., temporal aspect,
4. information about certain embeddings, text markers, or construction types,
5. hierarchical representations.

In an approach where post-head as well as pre-head recursion of NPs is allowed, neither the first nor the last noun must necessarily be the head of the NP. In order to extract the correct head of a structure, head lemma annotation is necessary. In (1.4a), e.g., the head of the NP *Fährschiff* is neither the first, nor the last noun.

Morpho-syntactic information is necessary to determine, e.g., a preference to use the plural for the A+N combination *falsche Hoffnungen* (false hopes), whereas for the A+N combination *stille Hoffnung* (silent hope), the singular is preferred. Besides, in order to extract N+N collocations that are in the relation of head to genitive modifier as in (1.4b), case information is necessary.

(1.4) a. [\text{Das von } [\text{Fährschiff } \text{‘Sassnitz’}] \text{ kommende } \text{‘the ferry ‘Sassnitz’ which is coming from Trelleborg (Sweden)’}]

b. [\text{eine gewisse Faszination } [\text{des Schattens }]] \text{ a certain fascination of the shadow} 

Lexical or semantic information is necessary to exclude or particularly include structures with a specific aspect. Noun phrases with temporal aspect (henceforth temporal NP) are usually not part of subcategorization frames, and named entities are usually not relevant with respect to collocations. In (1.5), e.g., there are in effect three NPs. The subcategorization frame of the verb *besuchen* (to visit), however, includes only two NPs, *Lisa* and *ihre Großmutter*. The third NP *jeden Tag* is a temporal adverbial and does not participate in the subcategorization frame. The only possibility to ascertain this is the temporal...
1.3 Requirements

aspect of the head noun Tag (day). Thus, only if the temporal aspect of the NP jeden Tag is marked, can the subcategorization pattern of besuchen be correctly assigned.

(1.5) \[ NP \text{Jeden Tag}\] besucht \[ NP \text{Lisa}\][NP ihre Großmutter ].

Each day visits Lisa her grandmother.

‘Lisa visits her grandmother each day.’

Information about certain embeddings, text markers, or construction types can provide additional information about properties of a structure or can indicate reliable access to information that can otherwise only be extracted out of mostly ambiguous contexts. The quotation marks surrounding the noun Sassenitz in (1.4a) indicate that the noun may be a description or name for something. If it follows a noun, it can be assumed to modify it.

Hierarchical representations provide additional linguistic and lexicographic information. The hierarchy in (1.4b), e.g., makes it clear that the NP des Schattens (the shadow) is a genitive modifier of the preceding noun Faszination (fascination), as it is embedded in the NP of which Faszination is the head. The NP ihre Großmutter (her grandmother) in (1.5), however, is not embedded in the NP of which the preceding noun Lisa is the head. Consequently, it is not a modifier of this noun.

1.3.3 Criteria for selection

The question is which depth of analysis best fits the needs for the application of extraction queries. The advantage of performing the analysis only online is modularity and theory independence. Small specific queries can be formulated easily without having to think about a possible representation of the constructions in an annotation scheme. Especially, if the phenomenon is not represented or compatible with the theory underlying the off-line analysis, an approach based only on online queries can be desirable. An online analysis can also be efficient if only one or a few simple queries have to be applied to a special corpus. In this case, a prerequisite analysis does not seem necessary.

However, if it is predictable that the corpus will be used for more than just a few simple extraction queries, or if the queries deal with complex phenomena, a previous off-line analysis seems to be desirable if not necessary. Analysis for general structures have to be performed only once. The results are annotated in the corpus. The extraction queries have access to this annotation, and do not have to perform the general parts of the analysis over and over again. The extraction process is thus faster.

A off-line annotation usually is more general and less restricted. The application of constraints and the larger rule system allows to cover more and more complex constructions. The analysis is more detailed and covers more
phenomena. It can thus form the basis of several different extraction queries. The complex analysis leads to a better coverage and, consequently, to a better recall. As more information is made available by the annotation, the quality of the results can also be increased without great effort.

For simple phenomena involving only very local relations, and if only one or few queries have to be applied to a corpus an online approach seems sufficient. However, if more complex phenomena and constructions are involved, and if several queries are to be applied, it seems desirable to have the corpus analyzed and annotated before the actual extraction takes place. The best, richest and most complex, and usually most reliable (especially if manual checking is involved) basis is provided by a full parse. However, full parsers often lack robustness and/or provide ambiguous output. Besides, they often have a slow parsing speed, which makes working with large corpora tedious.

A chunk analysis, being robust, fast and considerably reliable seems a good alternative for large scale extractions.

1.4 Classical chunking

According to Steve Abney (Abney 1996a) a chunk is:

- a non-recursive core of an intra-clausal constituent, extending from the beginning of the constituent to its head.

Another definition by Abney (Abney 1991) is:

- The typical chunk consists of a single content word surrounded by a constellation of function words, matching a fixed template.

Gee and Grosjean (Gee and Grosjean 1983) give psychological evidence for chunks, defining them as structure of word clustering that emerges from a variety of experimental data, such as pause durations in reading, and naive sentence diagramming. They speak of $\phi$-phrases and define them as an input string broken after each syntactic head that is a content-word.

Thus, the classic notion of a chunk is that of a flat, non-recursive structure. The chunk begins with a function word and ends with the lexical head. This definition excludes all post-head complements and modifiers. Consequently, some chunkers consider a prepositional phrase (PP) to consist only of the preposition itself, as PPs, in general, are head initial structures.

1.4.1 State-of-the-art systems

CASS parser

The CASS parser has been developed within the CASS environment of Steven Abney. The rules are applied in finite-state cascades as described by Abney
1.4 Classical chunking

The German CASS grammar developed within the Verbmobil project produces flat, non-recursive structures that are within the chunk definition of Abney (Abney 1991) given above. The grammar includes a small lexicon which is represented using so-called tag-fixes (see the Manual of CASS (Abney 1997) for more details). This means that part-of-speech (PoS) tags can be associated with lexicon classes. In this case, nouns of different lexicon classes would have different PoS-tags indicating their lexical property. Information about the head lemma of chunks is annotated as an attribute of the chunk. A demo version of the German grammar can be accessed via Internet (Verbmobil Projekt).

Conexor

The Conexor system is a symbolic constraint grammar parser (Voutilainen 1994; Voutilainen and Järvinen 1995; Voutilainen and Tapanainen 1993). The system was primarily built for English and has a full-fledged grammar for English, the ENGCG (Karlsson et al. 1995; Voutilainen 1997; Voutilainen and Heikkilä 1994). For some other languages a full grammar has also been developed, for German there is only a light weight version of the grammar. The output of the system for German consists of simple, non-recursive structures. Lexical information is not available, however, the head lemma of chunks is indicated by a special tag. Unfortunately, the system is not freely available, only a demo version is accessible via the Internet (Conexor).

KaRoParse

The Cascading Robust Parser KaRoParse developed by Frank H. Müller and Tylman Ule (Ule and Müller 2001; Müller and Ule 2001) is a symbolic top-down bottom-up partial parser. The output is a partial analysis including pre-head but no post-head recursion. The internal structure of a chunk is, however, flat and non-hierarchical. The chunk definition of Müller and Ule states that a chunk extends from the beginning to its head only. Ambiguous attachment decisions are not made to avoid false analyses or the need for manual disambiguation. Agreement information as well as lexical information about the chunks is not available. Additionally to lexical phrases they annotate topological fields (Müller and Ule 2002; Veenstra et al. 2002) which divide a German sentence into different sections according to the topological field model of (Höhle 1986).

Chunkie

Chunkie is a partial parser that works with similar techniques as standard PoS taggers (Skut and Brants 1998). It uses the TnT tagger (TnT)(Brants 2000) to assign tree fragments to sequences of PoS tags. The most likely structure is

Conexor

The Conexor system is a symbolic constraint grammar parser (Voutilainen 1994; Voutilainen and Järvinen 1995; Voutilainen and Tapanainen 1993). The system was primarily built for English and has a full-fledged grammar for English, the ENGCG (Karlsson et al. 1995; Voutilainen 1997; Voutilainen and Heikkilä 1994). For some other languages a full grammar has also been developed, for German there is only a light weight version of the grammar. The output of the system for German consists of simple, non-recursive structures. Lexical information is not available, however, the head lemma of chunks is indicated by a special tag. Unfortunately, the system is not freely available, only a demo version is accessible via the Internet (Conexor).

KaRoParse

The Cascading Robust Parser KaRoParse developed by Frank H. Müller and Tylman Ule (Ule and Müller 2001; Müller and Ule 2001) is a symbolic top-down bottom-up partial parser. The output is a partial analysis including pre-head but no post-head recursion. The internal structure of a chunk is, however, flat and non-hierarchical. The chunk definition of Müller and Ule states that a chunk extends from the beginning to its head only. Ambiguous attachment decisions are not made to avoid false analyses or the need for manual disambiguation. Agreement information as well as lexical information about the chunks is not available. Additionally to lexical phrases they annotate topological fields (Müller and Ule 2002; Veenstra et al. 2002) which divide a German sentence into different sections according to the topological field model of (Höhle 1986).

Chunkie

Chunkie is a partial parser that works with similar techniques as standard PoS taggers (Skut and Brants 1998). It uses the TnT tagger (TnT)(Brants 2000) to assign tree fragments to sequences of PoS tags. The most likely structure is
determined using trigram frequencies. In contrast to most chunkers, Chunkie produces structures with a maximal depth of three. It includes recursion in pre-head position, whereas post-head modifiers are usually not considered. The structure is represented in hierarchical way. Head lemma information is available via the structure. Agreement information and lexical information is not annotated (Chunkie).

Cascaded Markov Models

The partial parser of Brants (Brants 1999) is meant as a pre-processing step for deep analysis. Similar to Chunkie it is based on PoS tagging techniques. The difference to a PoS tagger is that a sequence of terminal nodes can be replaced by one non-terminal node. The grammar consists of stochastic context free grammar rules and left-to-right transitional context information (context sensitive). It was used to facilitate the syntactic annotation of the NEGRA corpus (NEGRA Project ; Skut et al. 1997). It is based on cascaded Markov Models, and operates on several layers. The output of each layer serves as the input to the next higher layer. The system was inspired by other finite-state cascades, e.g., CASS (Abney 1991; Abney 1996b).

The annotated structures are hierarchical phrases, including complex and recursive embedding. PPs and NP structures are stripped off adverbials at the front, and PPs and relative clauses at the end of the phrase. Head lemma information is available via the structure. Agreement and lexical-semantic information is not available.

The partial parser of Brants does not follow the longest match strategy but takes the most probable sequence instead. It is even possible to select analyses from lower layers, if they fit better to the highest layer. The best path is found using the Viterbi algorithm (Viterbi 1967). For training, the system requires existing treebank data.

Schiehlen’s chunker

Another chunker for German which goes beyond mere base chunks is the chunker developed by Schiehlen (2002). The chunk definition follows that of (Brants 1999). It is based on symbolic context free grammar rules. It includes recursive embedding in pre-head and in post-head position, but again no PP-attachment or relative clause attachment. The head lemma, and lexical-semantic information is not available.

The chunker needs optimally tokenized text, which includes the recognition of (multi-word) named entities. A set-up which is not realistic for the application on large quantities of unknown text data, as we cannot expect to know all named entities and multi-word units included in such texts.
1.4.2 Problems for extraction

The classic notion of a chunk includes only relatively simple structures. Extraction queries applied to a corpus with a classic chunk analysis will have to combine the chunks to form larger structures. Chunks should, therefore, be easily and reliably combinable. However, as Kübler and Hinrichs (Kübler and Hinrichs 2001) have pointed out, while chunking approaches have

focused on the recognition of partial constituent structures at the level of individual chunks [. . . ], little or no attention has been paid to the question of how such partial analyses can be combined into larger structures for complete utterances.

In other words, combining chunk structures provided by most available chunkers often needs complex rules or rules that are neither certain nor theoretically motivated. For German, this is even more so than for English, for which most chunkers are designed. This is due to the fact that German has a tendency for complex phrase structures, and for pre-head embedding of complex structures.

If we take the following example of a PP, with its full analysis in (1.6a) and compare it with the classical chunk analysis in (1.6b), the problems for the extraction queries become obvious.

(1.6) a. \([_{PC} \text{mit } _{NP} [_{AP} \text{kleinen } ]], _{AP} \text{über } _{NP} \text{die Köpfe } _{NP} \text{der Apostel } ]\] gesetzten ] Flammen ]
apostles set flames

‘with small flames set above the heads of the apostles’

b. \([_{PP} \text{mit } _{NC} \text{kleinen } ]], _{PC} \text{über } _{NC} \text{die Köpfe } _{NC} \text{der Apostel } ] _{NC} \text{gesetzten Flammen } ]\]
apostles set flames

1. The chunk analysis provides four NCs where there should be only one NP.

2. An extraction of adjective noun collocations will deliver the pair gesetzten + Flammen but not the pair kleinen + Flammen. The former, however, is not a adjective noun pair but rather a verb noun pair. Knowing about the deverbal character of gesetzten, this pair can be excluded. The latter is a correct adjective noun pair, however, it can not be extracted in a straightforward way as the two tokens are not under the same NP node.

3. The candidate pair Köpfe + Apostel for noun noun collocation cannot be found, if the NC der Apostel is not identified as a genitive modifier of the...
Introduction

...preceding NC. This can only be done if agreement information is available.

4. The candidate pair *setzen + Flammen* for verb noun collocation can only be extracted if the deverbal character of the adjective *gesetzten* is indicated.

In order to extract the information contained, a more complex analysis than in (1.6b) and additional lexical information is needed. Therefore, the hierarchical structure in (1.6a) has to be assembled combining the chunk in (1.6b). Besides, the deverbal character of *gesetzten* has to be determined.

In order to fill the gap between the chunk analysis in (1.6b) and full analysis in (1.6a), the PCs and NC in (1.6b) have to be combined.

The simple solution

The simple solution would be a rule such as the following:

\[(1.7) \quad PP \rightarrow PC(PC|NC)^*\]

This rule, however, does not seem theoretically motivated. It might seem logical to assume that a complex PP consists of a PC followed by a number of NCs. It is, however, difficult to find a theoretical background to include following PCs as well. PPs usually do not directly embed another PP or PC. Besides, it is questionable that (1.7) is the only rule needed. Other structures might involve other elements on token or structure level that would also have to be included. We, however, cannot depend on the theory to find this out, as the theoretical motivation for these additional rules is probably also lacking.

The rule is rather vague and underspecified such that it does not seem very reliable. The rule can easily assemble structures which are too large, and it seems difficult, if not impossible, to formulate restrictions.

Besides, the rule leaves the internal structure mainly opaque. It is, e.g., not obvious what the head of the NP embedded in the complex PP is. The relation between the NC *die Köpfe* (the heads) and the NC *der Apostel* (the apostles) is not indicated. Consequently, the head can equally well be *Apostel* (apostles) or *Flammen* (flames). In the former case, the NC *gesetzten Flammen* (set flames) would be a post-head genitive modifier. In the latter case, the NC *der Apostel* (the apostles) would be a pre-head modifier. In order to extract all the information contained in (1.6a), the internal structure would also have to be made transparent. This, however, is not possible with simple rules such as (1.7). A more complex solution appears necessary.
The complex solution

First, the NC *der Apostel* (the apostles) has to be identified as a post-head genitive modifier. In order to do this, morpho-syntactic information about the chunk has to be available. Classic chunkers usually do not provide agreement information about annotated structures. If at all, morpho-syntactic features are available on token level. Therefore, this information has to be added.

Second, the deverbal character of the adjective *gesetzten* has to be identified. This lexical property is necessary to be able to treat the participle as a verb instead of an adjective with respect to the extraction of collocation or subcategorization information. Again, most state-of-the-art systems do not provide this kind of information in their annotation. Therefore, the use of external sources, e.g., ontologies, lexica, dictionaries or word lists is required to filter the extraction results.

Third, a number of rules have to be applied

(1.8)  

a. NP → NC NC_{gen} 

b. PP → preposition NP 

c. AP → PP adjective 

d. NP → AP* noun 

The NP rule in (1.8a) will assemble the NC *die Köpfe* with the NC *der Apostel*. The PP rule in (1.8b) can build the PP *über die Köpfe der Apostel* as well as the complex PP in (1.6, p.23). The AP rule in (1.8c) can be used to form a complex AP with this PP (*über die Köpfe der Apostel gesetzten*). The NP rule in (1.8d) can build the complex NP *die über die Köpfe der Apostel gesetzten Flammen*. Besides these rules, the NC *gesetzten Flammen* has to be split up, to be able to construct the complex AP *über die Köpfe der Apostel gesetzten* and finally, the complex NP in which it is embedded.

The rules listed in 1.8 are able to build the hierarchical analysis in 1.6a. For other examples, one would probably need other or additional rules. The number of rules needed for this single example shows that a classic chunk analysis leaves much of the parsing to further applications. If extraction queries are applied to a chunked corpus, the queries have to include a lot of linguistic knowledge in the form of rules. This, however, makes the queries complex and unwieldy. Thus, classic chunk structures do not seem to be appropriate to form a useful basis for extraction processes. In order to support extractions, the classic chunk concept has to be extended.

**CASS as an environment for chunking**

CASS is an environment developed by Steven Abney (CASS ; Abney 1997). It allows the easy development of simple grammars for chunking or partial
parsing. Recursive structures with finite depth can be built using finite-state cascades, such as, e.g., the CASS parser (Abney 1996b). Different levels of rules can be applied in cascades. The output of one level serves as input to the next level. Once a structure is annotated at a certain level, the content becomes opaque to the following levels. It is, therefore, not possible to access the PoS-tags of tokens, once they are a part of an annotated structure. The same holds for embedded structures.

It is also not possible to run one and the same level several times. The output of a level cannot be taken as input for the same level. The only possibility of repeating the rules of a level is to copy the whole level. In other words, in order to perform one or more rules several times in different cascades, the rules have to be copied to different levels. If a rule is changed or modified, these changes or modifications have to be done for each instance of the rule.

The CASS environment allows work with morpho-syntactic information, and the use of intersection to determine the agreement elements of an annotated structure. The head lemma and other lexical properties of the chunk can be annotated as feature attributes. A lexicon can be included by using so-called tag-fixes. Including lexical information via tag-fixes, however, implies changing PoS-tags. In other words, in order to express the lexical property of a token, its PoS-tag has to be changed. This, however, means that there have to be several PoS-tags for a single category, such as nouns or adjectives: A fact that is not desirable as the commonness of the categories is lost to a certain extent.

Complex rules involving aspects like agreement, and lexical properties are difficult to develop. The same holds for constraints. CASS cannot make use of regular expressions to formulate rules or constraints. An interactive grammar development is not possible, as in order to see the result of a new rule, or a change or modification of an existing rule, the whole process has to be run. It is not possible to test a rule on the corpus at the state in which it would be applied.

All in all, CASS seems to be a useful environment when there is a need to build a simple grammar in a short time. However, if the grammar is going to be more complex and if many associated aspects (head lemma, agreement, lexical features) are to be included, the grammar development becomes complicated and unwieldy. Besides, the fact that regular expressions cannot be used in the rules makes underspecifications of this kind impossible. A further shortcoming is the fact that annotated structures are opaque, i.e., their inner structure cannot be accessed any more.

### 1.5 Conclusion

An important requirement for a text analyzing tool that is to provide a useful basis for extraction processes is robustness. In other words, the tool has to be
1.5 Conclusion

capable of analyzing unrestricted text i.e.:

1. it must not be domain specific,

2. it has to work on small as well as on large corpora,

3. it has to be able to parse full sentences as well as fragmentary text,

4. ill-formed sentences should not result in a complete failure to parse

5. failure to parse parts of a sentence should not prevent the analysis of the rest of the sentence

Chunkers which perform a partial analysis seem best fit for this requirement. For a chunker there is basically no limitation with respect to the type and amount of text that is to be analyzed. The text neither has to consist of full sentences, nor does it have to be wholly grammatical. Chunkers usually work bottom-up. Chunk rules are not dependent on one another, consequently, the analysis of one chunk is independent of the analysis of other chunks in the sentence. Thus, if parts of a sentence cannot be analyzed this does not prevent the analysis of the rest of the sentence. If single tokens cannot be included in a larger constituent, they are left stranded.

There remains the question of grammar development and grammar type. Machine learning methods make us independent of a human grammar writer, and can exploit corpora as a knowledge source for the grammar rules. On the other hand, they depend on a training corpus, where the linguistic knowledge is encoded. Consequently, only the phenomena present in the training corpus are covered. This holds for the domain of the corpus as well. Hand-written rules have the advantage that one has a good control of them. As a chunker has a relatively simple grammar system, the grammar development is also less costly. Moreover, hand-written rules seem to be more capable of dealing with the requirements to annotate additional information about the properties of chunks than machine learned rules.

Similar reasons favor a symbolic grammar instead of a probabilistic one. Additional information about lexical, semantic and morpho-syntactic properties is easier to include in a symbolic approach than in a probabilistic one. Besides, probabilistic grammars are more restricted with respect to corpus size. Small as well as very large corpora can pose problems for a probabilistic grammar. Small corpora do not provide enough material for a statistical evaluation. Large corpora can exceed the working capacity of most machines. Another problem with respect to working capacity are sentences containing too many words. Besides, adaptation of the grammar to text domains is easier within a symbolic approach than within a probabilistic one.

To sum up, for the application as an analyzing tool in computational lexicography a symbolic chunker with a hand-written grammar seems to be a good
Introduction

choice. The available chunkers for German, however, do not consider all of the additional information needed for this task such as head lemma, morpho-syntactic information, and lexical or semantic properties, which are useful if not necessary for extraction processes. The CASS environment is not powerful enough to deal with these aspects in a efficient and transparent way. Thus, we decided to build a recursive chunker for unrestricted German text, YAC, within the framework of the IMS Corpus Workbench (CWB). Chapter 2 describes the general concept of YAC, the technical framework, and the application of the rules. It gives an overview of the annotated structures, and feature attributes. Chapter 3 describes the grammar in detail. It gives the annotation scheme, feature annotation, and grammar rules for each chunk category. Chapter 4 discusses the evaluation of chunkers in general and YAC in particular. Chapter 5 gives examples of possible applications.
Chapter 2

General Concept of Off-line Analysis

2.1 Introduction

YAC is a recursive chunker for unrestricted German text. It works fully automatically without the need for manual disambiguation or correction. The main goal is to provide a useful basis for the extraction of linguistic as well as lexicographic information from corpora. Consequently, the grammar rules of YAC are implemented so as to make the resulting analysis meet the needs of an ensuing extraction process.

YAC is based on a symbolic regular expression grammar written in the CQP query language (see section 2.3.2) which is part of the IMS Corpus Workbench (see section 2.3.1). The chunker works on a corpus which is tokenized and part-of-speech tagged using the STTS-tagset (Schiller et al. 1999). For tokenization and PoS-tagging the TreeTagger (TreeTagger; Schmid 1994; Schmid 1995) is used. The German grammar additionally requires lemma and agreement information on token level, which is annotated using the IMSLex morphology (Lezius et al. 2000; Lüdeling and Fitschen 2002).

2.2 A recursive chunker for unrestricted German text

In some respects YAC is a typical chunker:

- it is robust, i.e., it works on unrestricted text,
- it works fully automatically,
- it does not provide a full but only a partial analysis of text,
General Concept of Off-line Analysis

- it does not make highly ambiguous attachment decisions.

As has been argued before, there are certain requirements to the tool itself and the annotation in the corpus that make it necessary to extend state-of-the-art chunking. Thus, YAC is also different from other state-of-the-art chunkers:

- it extends the classic chunk definition of Abney
- it provides additional information about the annotated chunks

The classic chunk definition of Abney as a “non-recursive core of an intra-clausal constituent, extending from the beginning of the constituent to its head” (Abney 1996a) is extended by two main aspects:

1. recursive embedding
2. post-head embedding

The chunks annotated by YAC are still intra-clausal constituents, however, they are no longer non-recursive but include recursive embedding in pre-head (2.1a, Figure 2.11) as well as in post-head position (2.1b). Post-head recursion automatically implies that the chunk does not end with the head but may have modifiers in post-head position. These modifiers can but do not necessarily have to have the same category as the chunk itself. In (2.1c), e.g., an adverbial phrase is embedded in an NP in post-head position.

(2.1) a. \[[NP \text{ die kleinen, üb}er [NP \text{ die Köpfe der Apostel }] \text{ gesetzten Flammen } \]
   \text{‘the small flames set above the heads of the apostles’}

b. \[[NP \text{ die Köpfe [NP der Apostel ]}] \]
   \text{the heads of the apostles}

 c. \[[NP \text{ Jahre [AdvP später ]]}
   \text{year later}

There are, however, certain limitations with respect to post-head modification. YAC annotates only non-ambiguous constructions. Consequently, highly ambiguous attachment decisions, such as PP-attachment, are not made. Solving these ambiguities requires comprehensive lexical, linguistic, and context information, and in some cases world knowledge. The example in (2.2) shows two possible analyses of a sentence with regard to PP attachment.

---

1Tree diagrams of YAC annotations are presented in the TIGERSearch format (TIGERSearch; Lezius 2002; Lezius and König 2000).
2.2 A recursive chunker for unrestricted German text

Figure 2.1: TIGERSearch tree diagram of (2.1a)
The PP von Gabor Kiraly can belong to the preceding NP das Tor as in (2.2a). In this case, its the goal belonging to Gabor Kiraly that the shot came at. The sentence can, however, also mean that the shot at the goal was performed by Gabor Kiraly. In this case, the PP von Gabor Kiraly does not belong to the preceding NP das Tor as in (2.2b). In this case, both analyses are possible from the point of lexical and linguistic knowledge. World knowledge, that is knowing that Gabor Kiraly is a Hungarian goal-keeper, makes the analysis in (2.2a) more likely, but is still remains ambiguous. Only context information can determine the correct analysis, that is, we have to know who shot, or at which goal. Solving ambiguities like this goes beyond the means of a chunker. The same applies to long distance relations. Consequently, discontinuous elements are left separate.

The chunk definition of Abney is extended and reformulated as follows:

(2.3) A chunk is a continuous part of an intra-clausal constituent including recursion, pre-head as well as post-head modifiers but no PP-attachment, or sentential elements.

The extension of the range of chunks, however, is not the only extension. YAC annotates a variety of additional information for the chunks, most of which state-of-the-art chunkers do not annotate:

1. head lemma,
2. agreement information
3. lexical-semantic and structural properties. The lexical-semantic properties are introduced by small lexical lists.

Some state-of-the-art systems do annotate the head lemma, but do not make agreement information and other properties of the chunks available. For extraction purposes, this information, however, is important as has been argued in section 1.3.2.

The process architecture of YAC is divided into three different levels of annotation. Each level addresses a different task with respect to the annotation of chunks (see section 2.4.3 for more details).
2.3 Technical framework

2.3.1 The IMS Corpus Workbench (CWB)

YAC uses tools of the IMS Corpus Workbench (CWB) (Christ 1994; Christ and Schulze 1995; CWB). The CWB provides an environment for storage and querying of text corpora, which enables work with large corpora. It was originally designed for corpora annotated on token level only, i.e., with PoS-tags and lemma values. Later support for non-overlapping structural annotations was added. Originally, the support for structural attributes was conceived for the annotation of document structure (source files, paragraphs, sentences). These structural attributes, however, do not need a powerful mechanism for storing and querying. The single structures can be in a hierarchical relation, but recursion does not occur. Thus, real recursion is not considered within the CWB framework, and has to be simulated by giving recursively embedded structures different names (e.g., NP1, NP2).

2.3.2 The Corpus Query Processor (CQP)

The CWB has a powerful corpus query processor (CQP) (Christ et al. 1999). Within the CQP framework queries can be specified in terms of regular expressions over tokens and linguistic annotation. It allows complex expressions at the basic token level and on XML-tags (structural mark-up) such as:

- regular expression matching on token and annotation strings
- tests for membership in user specific word lists
- special operations on feature sets
- anchor points and labels
- constraints to specify dependencies

**Regular expression matching**  Regular expression matching on token level enables the search for specific realizations of annotations on token level, such as specific word forms, lemmas, PoS-tags. Regular expressions on string level permit values to be left underspecified. In order to include all the temporal nouns below in a query, it is sufficient to use the regular expression \( .* \text{jähr} \).

Frühjähr
Vorjähr
Geschäftsjähr
Halbjähr
Regular expression matching can also include XML-tags, which represent the structural annotations.

**Tests for membership in user-specific word lists**  
CQP also provides the possibility to test for membership in user-specific word lists. These lists can be either defined directly in a CQP session listing the items, or they can be read in from separate files which contain lists of items. The lists are stored in variables. Variable names in CQP start with the `@` symbol. When the query is executed the item at the respective position is compared to the items in the list. The query is only successful, if the item is a member of the respective list. The following rule for cardinal noun chunks searches for structures such as 480 Millionen (480 million), where the head lemma of the phrase is represented by a list ($noun_Card$).

```cqp
[pos = "CARD"]?
@?[lemma = RE($noun_Card$)]
```

Thus, the head lemma is restricted to nouns of a specific class. In this case, cardinal nouns, such as Million (million); Tausend (thousand). The head lemma is additionally marked by an anchor point (the target, `@`) to allow its identification. The operator `RE(...)` can process lists which can include regular expressions. The following displays the list for cardinal numbers, which includes both full lemmas and regular expressions of lemmas.

Billiarde
Billion
Dutzend
Milliarde
Million
Tausend
Trilliard
Trillion
  *billiarde
  *billion
  *dutzend
  *milliarde
  *million
  *tausend
  *trilliarde
  *trillion

The single items of the list are treated as disjoints.

**Feature set operations** CQP has the possibility of storing single value items as well as feature value sets. Feature sets are codes of string values with a specific disjunctive notation. The single values of feature value sets are separated by |. Thus, agreement information can be stored as single feature-set annotation of all possible combinations of case, gender, number, and definiteness values for a certain word or phrase. The single elements are separated by a colon.

```
|Akk:M:Sg:Def|Akk:M:Sg:Ind|Akk:M:Sg:Nil
|Dat:M:Sg:Def|Dat:M:Sg:Ind|Dat:M:Sg:Nil
|Nom:M:Sg:Def|Nom:M:Sg:Ind|Nom:M:Sg:Nil|
```

The single feature value sets can be queried with special operators. It can, e.g., be tested whether a feature set includes a certain value. Thereby, we differentiate between the operators `contains`, where the value has to be included in the feature set, and `matches`, where the value has to be the only value included in the feature set. The two operators `matches` and `contains` can be combined with regular expressions on string values. In combination with a regular expression the `matches` operator can be used to find feature sets where parts of the values of a feature set have to be common to all values. The following query part searches for feature sets including only genitive case.

(2.4)  [agr matches "Gen:*"]

The query can find all instances of feature sets, where all single values include the string `Gen:`. Thus, e.g., the following still ambiguous feature set:

```
Gen:N:Sg:Ind|Gen:N:Sg:Nil|
```
Apart from the two search operators, there is an operator that allows checking of consistency of two feature sets using (set-theoretic) intersection. It is possible to check whether feature values of tokens or XML-tags agree with each other. The result of the (set-theoretic) intersection can be additionally used to disambiguate morpho-syntactic information (see section 2.3.3 for more details).

Constraints and anchor points  CQP has two anchor points for a query: the beginning of a match (match) and the end of a match (matchend). Another anchor point (target) can be specified within the query using a specific marker (@). In addition to these labels, other labels can be explicitly set. In 2.5 the label a is set on the second position of the query.

(2.5)  \[[pos = "ART"] a: [pos = "ADJA"] [pos = "NN"]\]

The labels can then be used to formulate constraints. The example in (2.6) shows a constraint on the properties of an adjectival phrase.

(2.6)  \[:.ap_f not contains "invar" & _.ap_f contains "attr":]\]

Instead of the positional attributes word, lemma, pos, and agr, the example in (2.6) refers to a structural attribute (ap_f)\(^2\). The underscore is a special label indicating that the structural attribute is to be evaluated at this position.

The constraint in (2.6) is processed in advance of the following token, indicated by the colons at the beginning and end of the bracket. In other words, the constraint and the token are separated from one another, with the constraint preceding the token it refers to. As structural annotation is connected to tokens, both the annotation on token and on structural level can be tested this way. It is also possible to formulate constraints within a token, where constraint and token are incorporated in one item as in the following rule for agreeing adjective phrases:

```
<ap>
  [  (._.ap_f not contains "invar" & _.ap_f contains "attr")
    -> (._.ap_suff = $0.ap_suff) ]
  []*
</ap>
```

In this case, constraint and token are merged, and the constraint does not refer to the following token. Constraints can be used as simple specifications or restrictions on structural attributes, i.e., they specify or restrict a certain feature attribute. Constraints, however, can also be used to specify dependencies within

---

\(^2\)The annotated structural attributes are more described in chapter 3 in more detail
or among arbitrary tokens. In the example query above, there is a constraint on the feature value \texttt{ap\_suff} of the adjectival phrase. The constraint states that the feature value of this adjectival phrase has to be equal to the feature value \texttt{ap\_suff} of another adjectival phrase, which is referred to by the label \$0\$. The constraint, however, depends on another constraint, i.e., only if the first constraint is fulfilled, does the constraint on the feature value \texttt{ap\_suff} have to be fulfilled. The dependency is marked by the operator \texttt{-\rightarrow}. Based on the query above, the feature values \texttt{ap\_suff} have to be equal only if the feature value set \texttt{ap\_f} of the respective adjectival phrase contains the value \texttt{attr} (attributive), and does not contain the value \texttt{invar} (invariant).

**Macro language**

Important for the use of CQP in a text analyzing tool is that CQP provides a simple macro language which is based on string replacement with interpolation of up to 10 arguments. Queries can be stored as query macros in separate files. This means that the rule system can be stored in files. The language for the query macros is the same as for regular CQP queries. Within a macro file, however, the queries can be formatted and commented to make them more transparent. Below we give an example of a simple macro for an NP structure consisting of a determiner and a noun.

```
MACRO np(0)
(
    [pos = "ART"]
    [pos = "N.*"]
)
;
```

The first line of each macro begins with \texttt{MACRO} followed by the name of the macro, and the list of possible arguments in parentheses. In the case that there are no arguments for the macro a 0 is given. The query itself is enclosed in parentheses, which have to be on separate lines. The end of each macro is marked by a separate line containing a semicolon.

Macros can also include other macros. Thus, complex rules can be broken up into smaller parts. This can be helpful if a certain pattern is used in more than one rule. In this case, the pattern has to be written only once, and if it has to be modified or adapted, this also has to be done only once. The included macros begin with a slash (/) followed by the macro name and possible arguments in parentheses. The grammar code below shows the rule for coordinated adjectival phrases including the macro for agreeing adjectival phrases.

```
MACRO cap(0)
(
```

37
General Concept of Off-line Analysis

<ap>
  a: [_.ap_f contains "attr"]
  []*
</ap>
/ap_agrees(a)*

([pos = "KON" | word = ",",]? /ap_agrees(a) + )*

The wildcard operator following the macro `ap_agrees` is applied to the whole macro content, whether it is only a single token or a whole code sequence. In the case of the macro for agreeing adjectival phrases, the label of the preceding AP (a) is handed over to the macro, so that the two APs can be compared. The rule for coordinated APs (cap) is used in noun chunk rules to include a principally unlimited number of adjectival phrases which agree among each other and which can optionally be coordinated. In other words, the macro cap is again included in another macro query.

2.3.3 Applying and processing the rules

While the actual grammar rules of YAC are written in the CQP language, CQP is invoked by Perl-scripts, which are also responsible for the processing, and the annotation of the rules. In other words, the Perl-scripts build a framework for the system. This has the advantage that in addition to the tools provided by the CWB, YAC can use all operations provided by the script language Perl as well. Figure 2.2 gives an image of the interactions among cqp and Perl-scripts.

![Interaction among CQP and Perl-scripts](image)

Figure 2.2: Interaction among CQP and Perl-scripts

The Perl-scripts are responsible for the following:
• the invocation of CQP
• the processing of the results, and
• the annotation of the results into the corpus.

For each level of annotation a separate Perl-script exists. It loads the relevant parameters and applies the rules for the respective level. The rules for each chunk category are grouped into rule blocks, and are applied in a sequence specified within the Perl-scripts (see section 3.1). Following Abney, the “longest match strategy” is used, i.e., only the longest matching string for each rule is taken.

After each rule has been applied to the corpus, the results of the query rule are handed back to the Perl-script for further processing. The results of the query comprise the following corpus positions:

• the beginning of the match,
• the end of the match, and
• optionally two additional positions:
  – the target position, which marks a position specified within the match
  – the keyword position, which is set after the query has been executed, and does not necessarily have to be within the range of the query match

We need the corpus positions to determine start and end point of the structure that is to be annotated, and to extract relevant information already annotated at certain positions. Within the framework of YAC the target position is, in general, used to mark the position of the head lemma. Marking the head position with a target is necessary as heads do not have a fixed position within the match. The keyword position can be used to mark a position inside and outside of the match. Keywords are not as reliable as targets, as they are specified only in relation to another position. During the parsing process it is only used in a few exceptional cases to include contextual information.

The resulting corpus positions can be used to access positional and structural attributes annotated in the corpus. All information stored on token or on structural level can be extracted, i.e., every annotated positional or structural attribute can be accessed. For example the lemma at the head position can be extracted, or the morpho-syntactic information of tokens and chunks relevant to determine the agreement information of the structure.

The extracted information can be further processed by the Perl-scripts:

• extracted values can be checked
General Concept of Off-line Analysis

- extracted values can be changed
- extracted values can be compared
- the range of the structure can be changed

**Checking values for membership in a specific list** If the lexical property of a head does not trigger a specific internal structure, it seems more reasonable to determine the lexical class of a chunk after rule processing. In this case, the same rule can be used for chunks with different lexical properties but the same internal structure, instead of having to multiply the rule for each different lexical head. The head lemma is then compared to entries in a list, and the respective membership is marked by a lexical attribute annotated in association with the chunk. The default AdvP rule AdvP, e.g., leaves the adverbial head underspecified with respect to its lexical-semantic properties. After the rule has been executed the heads of the resulting structures are compared to lists of different adverb classes. If a head lemma is included in one of the lists, the respective feature (e.g., Voc) is annotated as feature attribute (see section 3.2.1 for more detail).

**Changing or modifying values** In the case of prepositions conflated with an article the annotated lemma is a contraction of preposition and article. For the head lemma annotation we want the bare preposition. The Perl scripts allow to trace the annotated lemma back to the preposition using a set of rewrite rules as in (2.7a+b).

(2.7)  
\[
\begin{align*}
\text{a. } \text{ins} &\rightarrow \text{in das} \\
&\rightarrow \text{in the} \\
\text{b. } \text{vom} &\rightarrow \text{von dem} \\
&\rightarrow \text{of the}
\end{align*}
\]

Another example is the verb lemma of verbs with separable prefixes. If the prefix is separated from the verb, the annotated lemma of the verb form no longer includes the prefix. We want the head lemma of verbal complexes to include both the bare verb lemma and the separable prefix. Thus, the annotated verb lemma is joined with the annotated lemma of the separated prefix to form the full head lemma as is exemplified in (2.8a+b).

(2.8)  
\[
\begin{align*}
\text{a. } \text{kommt an} &\rightarrow \text{ankommen} \\
&\rightarrow \text{to arrive} \\
\text{b. } \text{gehen weg} &\rightarrow \text{weggehen} \\
&\rightarrow \text{goes away}
\end{align*}
\]
Comparing values  In the case of agreement, the morpho-syntactic information of relevant elements of the respective structure is compared, and a (set-theoretic) intersection is made, in order to disambiguate the agreement information. If the (set-theoretic) intersection returns a value, it is annotated as the morpho-syntactic information of the whole structure. If there is no result, i.e., if the single elements of the structure do not agree, the structure is not annotated at all. Figure 2.3 displays the morpho-syntactic information of the single elements of the NP den vierten Tag (the fourth day). It can be observed that

den
|AkK:M:Sg:Def
vierten
|AkK:M:Sg:Def|AkK:M:Sg:Ind|AkK:M:Sg:Nil|AkK:N:Pl:Def
|AkK:N:Pl:Ind
|Dat:M:Sg:Def|Dat:M:Sg:Ind|Dat:N:Pl:Def|Dat:N:Pl:Ind
|Dat:N:Pl:Nil|Dat:N:Sg:Def|Dat:N:Sg:Ind
|Gen:N:Sg:Ind|Gen:N:Sg:Nil
Platz
|AkK:M:Sg:Def|AkK:M:Sg:Ind|AkK:M:Sg:Nil
|Dat:M:Sg:Def|Dat:M:Sg:Ind|Dat:M:Sg:Nil
|Nom:M:Sg:Def|Nom:M:Sg:Ind|Nom:M:Sg:Nil

Figure 2.3: Morpho-syntactic information of the elements of the NP den vierten Tag

...
Changing the range of a structure  In some cases, start and end point of the match are not equal to start and end point of the chunk. Thus, rule match, and actual chunk have a different range. The reason is that context information is included in the rule. Complex APs, e.g., can embed PPs in pre-head position, if the AP is embedded in a NP with at least one element preceding the AP inside the NP (see section 3.3.1 for more details). The rule for these APs returns structures as in (2.9).

(2.9) a. des \([AP\text{ zu Tode genudelten }\]
the to death played
b. wechselnden , \([AP\text{ zu Schatten aufgelösten }\]
changing , to shadows dissolving

The possibility to change the range of the structure within the Perl-scripts, in this case, to cut off a specific element, allows for underspecifications. Context information can be included to exclude unwanted structures. As can be observed, the structures (2.9) include context elements as well. Thus, the structure of the AP has to be reduced by these element in order to annotate the correct range of corpus positions for the chunk. Within the Perl-scripts the superfluous elements are identified, and the start position is reset.

2.3.4 Output formats

YAC works with two different corpus formats, which serve different purposes during and after the annotation process:

- the CQP internal format, used for
  - interactive grammar development
  - parsing
  - extraction

- an XML format, used for
  - hierarchy building
  - extraction
  - data exchange

CQP is the basis for an interactive grammar development allowing the grammar writer to develop rules by applying them directly to the corpus at the relevant stage of annotation. The rule is tested and modified until the desired results are obtained. The chunking process is continued with the new or modified rule. During the chunking process the rules of YAC are applied to a CQP corpus. Intermediate results of the chunker are annotated into the CQP corpus.
Subsequent rules can be applied to this intermediate corpus. As CQP does not support recursion as such, intermediate results of the annotation process are collected in XML encoding. After the chunking has been completed the collected structures are used to build the full hierarchy (see section 2.4.3 for more detail). CQP is one of the two output formats of YAC. It is used for interactive querying of the annotated corpus.

XML is also used as an output format. It can be queried using XSLT style-sheets. An example sentence with XML output is given in Figure 2.4. As XML is a standard encoding format, it can be used for data exchange with other tools such as, e.g., TIGERSearch (TIGERSearch, Lezius 2002). The XML output-format of YAC, however, is not equivalent to the TIGER-XML format as described in (Lezius et al. 2002). It is based on a flat annotation of structural attributes. Non-terminal nodes are represented as XML-elements, which embrace embedded elements (terminal and non-terminal). We choose to represent non-terminals as XML-elements instead of as attributes because the same format without the token elements is used for the write-back to the CQP format. CQP, however, can only deal with non-overlapping structures. As we do not use a DTD, new XML-elements, i.e., new non-terminal nodes, can be added, as well existing XML-elements omitted or changed. Besides, as the syntactic annotation produced by YAC is flat, relationships between elements do not have to be expressed. A more elaborate XML-format as proposed, e.g., by (Mengel and Lezius 2000) is not necessary. On the contrary, it is not desired as it makes processing with XSLT-style-sheets for extraction processes unwieldy.

2.3.5 Advantages of the architecture

Using an efficient query language formalism for the grammar rules, and a separate post-processing by Perl-scripts has the following advantages:

- efficient work even with large corpora
- modular query language
- interactive grammar development
- powerful post-processing of rules

Powerful compression algorithms allow work with large corpora. Even complex queries can be efficiently evaluated and processed. It is possible to work with corpora of 200-300 million tokens. In order to keep working memory to a size feasible even for regular PCs, the corpus can be split into slices of variable size. We usually work with a slice size of about half a million tokens. The splitting of the corpora allows a principally unlimited corpus size.

The query language is modular, i.e., it allows the splitting of complex rules into different blocks. The building blocks can be combined into a variety of
General Concept of Off-line Analysis

Figure 2.4: Example sentence in the XML-output-format
different rules. General rule blocks have to be formulated only once, and can be used for an easy treatment of special cases. Only the deviant part of the query has to be newly formulated, the general block can be included as a macro template.

The fact that the grammar rules are written in a query language allows an interactive development and testing of the rules in the sense of rapid prototyping. The parsing process can be interrupted at any time in order to test new or changed rules on the current state of the corpus in an interactive CQP session. In other words, the grammar rules can be applied to the same corpus stage they are applied to during the parsing process. The results of the rule can be viewed on the screen. If necessary, the rule can be modified and applied again until it provides the desired results. After testing, the parsing process can be continued with the new or modified rule.

The same formalism used for the grammar rules can be used for interactive querying of the final results, i.e., the same formalism used for the rule system, can be used to extract linguistic as well as lexicographic information. There is no conceptual difference between extraction queries and annotation rules other than their function. No specific query language has to be learned, and no complicated interface is necessary to provide access to the annotated data. Besides, extraction templates covering structures which are found to be relevant for more than one extraction, can be easily included in the parsing process if desired. In this case, the extraction query is simply taken as a grammar rule. It is applied together with the other rules, and the result is annotated in the corpus. As corpus annotation it is easily accessible for further extraction processes.

The use of Perl-scripts as a framework for the rule system allows a powerful post-processing of the results of the grammar rules. Grammar rules can be left unspecified, as the Perl-scripts have control over the results. The Perl-Scripts can filter, change, and reject the results if necessary. Context information can also be included to a certain extent, as the range of a structure can be changed. Lexical information can easily be included without multiplying the grammar size unnecessarily. Different output formats can be provided, and hierarchical structures can be built.

## 2.4 General framework of YAC

### 2.4.1 Linguistic coverage

The structures annotated by YAC comprise the following lexical phrase categories:

1. adverbial phrases (AdvP)
2. adjectival phrases (AP)
3. noun phrases (NP)
4. prepositional phrases (PP)
5. verbal complexes (VC)
6. single verbs (V)

Within the framework of YAC two different groups of phrasal categories are differentiated:

- simple phrasal categories, and
- complex phrasal categories

Simple phrasal categories are limited with respect to their internal structure. They either do not embed other structures at all, or embed only simple structures, i.e., structures which themselves do not embed other structures. Adverbial phrases (AdvP), e.g., either do not embed any other structure, as in (2.10a), or they embed simple other AdvPs, as in (2.10b).

(2.10) a. vielleicht (perhaps)

   b. “sehr bald” (very soon)

In contrast, complex phrasal categories have the potential for a complex internal structure. They can embed other structures even if the embedded structure is itself complex. (2.11) gives an example of a complex PP in which a complex NP is embedded. The NP itself embeds two APs, the single-item AP kleinen, and the complex AP über die Köpfe der Apostel gesetzten. The latter on its parts embeds another PP über die Köpfe der Apostel. The PP embeds another NP die Köpfe der Apostel, which itself recursively embeds another NP der Apostel.

(2.11) mit [NP [AP kleinen ], [AP über [NP die Köpfe [NP der Apostel ]] gesetzten ] Flammen ]]

‘with small flames set above the heads of the apostles’

The complexity of the structure is illustrated in the TIGERSearch tree diagram in Figure 2.5.
2.4 General framework of YAC

Figure 2.5: TIGERSearch tree diagram of (2.11)
Adverbial phrases

Adverbial phrases are relatively simple constructions consisting of an adverb and an optional particle. In certain contexts, an optional adverbal modifier is allowed. Examples of possible AdvPs are given in (2.12).

(2.12) vielleicht (perhaps); zu bald (to soon)

Adjectival phrases

Adjectival phrases can be simple as well as complex phrase structures. The most simple APs consist solely of the adjective head as in (2.13a). Slightly more complex are APs embedding adverbial structures as in (2.13b). Other APs can comprise specific constructions triggered by a certain lexical head as in (2.13c), and complex embeddings as in (2.13d).

(2.13) a. möglich (possible)
    b. schreiend lila
       screamingly purple
    c. rund zwei Meter hohe
       around two meters high
    d. über die Köpfe der Apostel gesetzten
       above the heads of the apostles set
       set above the heads of the apostles

Noun phrases

Noun phrases range from simple constructions consisting of a single noun or pronoun as in (2.14a), specific constructions as in (2.14b–c) to complex constructions involving recursion in pre-head positions as in (2.14d).

(2.14) a. Oktober (October); er (he)
    b. 4,9 Milliarden Dollar
       4.9 billion dollar
    c. "Frankensteins Fluch"
       Frankenstein’s curse
    d. kleine, über die Köpfe der Apostel gesetzte Flammen
       small, above the heads of the apostles set flames
       small flames set above the heads of the apostles
2.4 General framework of YAC

Prepositional phrases

Prepositional phrases comprise pronominal adverbs as in (2.15a) and more complex structures embedding coordinated NPs as in (2.15b) as well as complex NPs as in (2.15b–c).

(2.15) a. davon (thereof)
   b. zwischen Basel und St. Moritz
      between Basel and St. Moritz
   c. mit kleinen über die Köpfe der Apostel gesetzten Flammen
      with small above the heads of the apostles set flames
      with small flames set above the heads of the apostles

Verbal complexes

Verbal complexes are simple structures comprising of adjacent verbal elements as in (2.16).

(2.16) a. gemunkelt (rumored)
   b. muß gerechnet werden
      has counted to be
      has to be counted
   c. zu bekommen
      to get

Clauses

Aside of the lexical phrase categories clausal structures (CL) are annotated. CL structures include finite (2.17a), and infinite clauses (2.17b+c), as well as relative clauses (2.17d).

(2.17) a. daß selbst Ravel sich amüsiert hätte
      that even Ravel himself enjoyed would have
      ‘that even Ravel would have enjoyed himself’
   b. Instrumente selbst zu bauen
      instruments themselves to build
      ‘to build instruments themselves’
   c. um einen Kaffee zu trinken
      in order a coffee to drink
      ‘in order to drink a cup of coffee’
d. . . , die man in der griechischen Tragödie findet.
   . . , which one in the Greek tragedy finds.
   ‘. . , which one finds in the Greek tragedy.’

2.4.2 Feature annotation

Feature attributes specifying certain properties and characteristics of the chunks are annotated as well. The properties are classified and stored in different feature attributes to ease access. One large disjunctive feature attribute holding all information would be unwieldy. As each chunk category has different characteristics, the annotated feature attributes vary from chunk category to chunk category. In other words, each chunk category has its own annotation scheme. For AdvP, e.g., only the head lemma and lexical properties are annotated, while for NPs the head lemma, lexical properties, and morpho-syntactic information are annotated. Table 2.1 gives an overview of the feature annotation for all chunk categories.

<table>
<thead>
<tr>
<th>feature value</th>
<th>AdvP</th>
<th>AP</th>
<th>NP</th>
<th>PP</th>
<th>VC</th>
<th>CL</th>
</tr>
</thead>
<tbody>
<tr>
<td>lexical-semantic properties (f)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>head lemma (h)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>corpus position of head lemma (hpos)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>agreement information (agr)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>last letter of head lemma (suff)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>verbal head lemma (vlem)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Feature annotation for all chunk categories

Head lemma

The head lemma of the chunk is taken from the lemma attribute at the head position. The head position is either specified in the rule using a target, or in the rule processing using a “fixed” corpus position, i.e., a position that can be determined independently of the actual results relative to another position. Normally, the head lemma is a single token, derived from a single corpus position. In some cases, however, the lemmas of several tokens have been subsumed to form the head lemma. Multi-word proper nouns, e.g., have a multi-token head lemma, as a single lemma cannot be filtered out. The head lemma of verbal complexes with separated prefixes is a single-token head. However, it has been taken from two different corpus positions. The head lemma of PPs consists of two separate head lemma items: the lemma of the preposition, and the lemma of the embedded NP.
Morpho-syntactic information

As has been argued before, YAC gains morpho-syntactic information of chunks using the morpho-syntactic features of relevant elements of the chunk. Invariant elements (e.g., invariant APs such as lila (purple)) are not considered. The morpho-syntactic information does not have to be, and in most cases, is not unique. Figure 2.3 (p. 41) shows the ambiguous morpho-syntactic information of the relevant elements of the NP den vierten Tag (the fourth day). If there is more than one element relevant for agreement, it is possible to reduce the ambiguity. An intersection of the different value-sets is used to determine the morpho-syntactic information of the chunk. In contrast to probabilistic approaches, no guessing is involved, i.e., if the value is still ambiguous, it is left ambiguous. In cases where no value is returned, i.e., the relevant elements do not agree, the chunk is rejected as agreement is required within a chunk.

Lexical-semantic properties

Lexical-semantic and structural properties are important for the parsing as well as for extraction processes. The properties can be triggers for specific internal structures, functions, and usages of chunks. Some of the properties are inherent in the corpus itself, i.e., they can be determined from the information already present in the corpus:

- PoS-tags
- Text markers

Named entities, e.g., can be derived from the PoS-tag NE for proper noun (2.18).

(2.18) \([_{NP} \text{Johann Sebastian Bach}]\)

    NE   NE   NE

Text markers such as quotation marks, parentheses, and brackets indicate the special character of a chunk (e.g., as named entity or possible modifier) (2.19), and can function as a secure context, so that the restrictions on the chunks can be loosened.

(2.19) "Wilhelm Meisters Lehrjahre"

Other properties are determined by external knowledge sources, such as lexica, ontologies, etc. Lexical and semantic properties, such as, e.g., locality for adverbs have to be introduced by word lists (2.20).

(2.20) hier (here); dort (there)
Another possibility to derive properties is from the chunking process itself. In this case, specific embeddings are indicated as properties of the embedding chunk. Complex AP embedding PPs (2.21a) and NPs (2.21b) are marked by a respective feature indicating the embedded structure.

\[(2.21)\]
\[a. \quad [\text{\textit{AP} } [\text{\textit{PP, über die Köpfe der Apostel } }] \text{gesetzten }] \quad \text{\'set above the heads of the apostles\'} \]
\[b. \quad [\text{\textit{AP} } [\text{\textit{NP, der } "Inkatha"-Partei }] \text{angehörenden }] \quad \text{\'belonging to the Inkatha-Partei\'} \]

Thus, in the internal representation, APs embedding PP are marked with the feature \textit{pp}, whereas APs embedding NPs are marked with the feature \textit{np}.

### 2.4.3 Chunking process

The chunking process is divided intro three phrase levels and one clause level, which serve different purposes:

- **First Level**
  - lexicon and base chunk level\(^3\)

- **Second Level**
  - main parsing level

- **Third Level**
  - finishing level

- **Clause Level**
  - annotation of clausal structures

Figure (2.6) illustrates the architecture of the parsing process of YAC. The first and the third level are applied only once, and the results are immediately annotated into the corpus. The second level is applied iteratively, the annotation of each iteration serving as input to the next run. The first level introduces the lexicon and annotated base chunks.

There are several advantages of annotating base-chunks with specific internal structures and introducing lexical and semantic information in the first level:

- the specific rules do not interact with the main parsing rules

---

\(^3\) A base chunk refers to a chunk in the classic sense as defined by Abney.
2.4 General framework of YAC

Figure 2.6: Parsing Architecture of YAC

- the rules for chunks which do not involve complex (recursive) embedding have to be applied only once
- the additional rules which are necessary to cover specific phenomena of specialized text domains can be included easily without affecting the main parsing process
- the rules of the main parsing process can be kept relatively simple and general, as most special cases are already covered\(^4\)
- only a relatively small number of "general" rules is needed for the main parsing process

First Level

In contrast to the second level, the first level runs only once. It can be seen as a preliminary stage for the main parsing process. As such, it serves several purposes:

1. it annotates base-chunks

\(^4\)This corresponds to the common approach of dealing with exceptions first.
2. it annotates chunks with a specific internal structure

3. it introduces lexical-semantic properties

Base-chunks are kernels of phrasal structures or small non-recursive chunks. The chunks are used as basis for the further annotation process.

Chunks with specific internal structures are base-chunks which do not follow the general rule pattern of their category. Consequently, they need special rules which are valid for a certain subclass of words only. A specific internal structure is usually triggered by the lexical or semantic properties of the head. In other words, the properties of the head have an impact on its behavior, i.e., the head can select specific modifiers or can build multi-word units. Temporal nouns, e.g., can take specific noun modifiers in pre-head position (as in 2.22a) and specific adverbial and year dates in post-head position as in (2.22b). Adjacent proper nouns (NE) can be subsumed to named entities as in (2.22c).

(2.22) a. Ende September; Mitte Januar
   end of September; mid of January

b. Jahre später; Juli 1999
   years later; July 1999

c. Johann Sebastian Bach NE NE NE

The dependencies and relations within these chunks are local, i.e., they do not involve long distance relations or complex (recursive) embedding. Thus, it is sufficient to apply these rules only once.

Apart from the base-chunks and the chunks with specific internal structure, lexical and semantic information is introduced in the first level. The annotated base-chunks are enriched with the properties of their heads. The lexical information can be introduced in two different ways:

- within the rules itself
- within the Perl-scripts

If the lexical property triggers a specific internal structure, it is included directly in the respective rule in the form of a word list. Otherwise, the word lists are compared to the lemma of the head after the general rules for the respective category have been executed. If the lemma is a member of one of the word lists, the lexical information is added as features associated with the chunk.

Second Level

The second level is the main parsing level. It includes relatively general and simple rules for each phrasal category, which allows complex (recursive) embedding. The rules in (2.23) schematize potential second level rules.
2.4 General framework of YAC

The AP rule in (2.23a) consists of an optional AdvP, an optional PP, and an AC as kernel of the phrase. (2.23b) shows an NP rule including an optional determiner, an optional cardinal, an optional AP, and an NC as kernel. The PP rule in (2.23c) contains a preposition and an NP.

The rules are applied iteratively to build up larger and larger structures which can involve complex (recursive) embedding. Thereby, the complexity of phrases is achieved by the embedding of complex structures rather than by complex rules. The same rules that build simple chunks and phrases can build complex phrases. The difference is simply the complexity of the embedded structures. In (2.24a–b), e.g., the only difference is that the AP in (2.24b, Figure 2.7) is more complex than in (2.24a). For both structures, the rule in (2.23b) has been used.

The same holds for the examples in (2.25a–b), the only difference is the complexity of the embedded NP. However, for both structures the same rule has been applied, namely the PP rule in (2.23c).

The rules of each phrasal category are grouped in rule blocks. Each rule block is applied in a separate CQP session. Thus, the results of each rule block are immediately available to the following rule blocks. This speeds up the process of annotation considerably, each rule block can not only build on the output of previous levels or runs, but also on the output of previous rule blocks. Thus, we need fewer iterations. An iteration needs considerably more time than closing and opening a new CQP session.
In contrast to the CASS environment (as described in 1.4.2), annotated structures are transparent, i.e., not only the output of the last iteration but the output of all previous runs is available. Only embedded recursive structures are not available to newly applied rules. In other words, only the largest version of an annotated structure is available. This is due to the fact that CQP does not allow overlapping structural attributes of the same category. All other embedded structures and all positional attributes, however, are available.

Previously annotated structures can form the kernel of a phrase, i.e., take up the head position as in (2.26a). Besides, annotated structures can also function as adjunct or complement of a phrasal structure as in (2.26b).

\[(2.26)\]
\[a. \quad [_{NP} \text{die ebenfalls 24jäh} \text{rige} \quad [_{NC} \text{Christina Siegl }] ] \]
\[b. \quad [_{NP} \text{die Köpfe} \quad [_{NP} \text{der Apostel }] ] \]

Thereby, the feature attributes specifying the lexical and semantic properties can be ignored during the parsing process, i.e., the chunks can occupy a position or fulfill a function independently of their associated features. Any NC, e.g., can build the kernel of an NP. The lexical and semantic properties, however, can also be used to put restrictions to a certain position. The post-head modifier position of NPs, e.g., is restricted to NPs or NCs, which either have genitive case, as in
2.4 General framework of YAC
(2.27) a. \( [_{NP} \text{die Probleme} \ [_{NP} \text{unseres Landes } ] ] \)
   the problems of our country

b. \( [_{NP} \text{das Imperium} \ [_{NP} \text{des “Tankerkönigs”} \ [_{NP} \text{Aristoteles} \ [_{NP} \text{Onassis } ] ] ] \]
   the empire of the “tanker king” Aristoteles
   Onassis

  ]]

c. \( [_{NP} \text{dem Slogan} \ [_{NC} \text{“The Windows Solution” } ] ] \)
   the slogan “The Windows Solution”

In contrast to many other approaches, YAC allows to consider special cases, i.e., in combination with specific contexts, the restrictions that are opposed to the rules, can be weakened. In combination with parentheses and quotation marks, e.g., the restriction on post-head modifiers can be loosened to include PP-attachment. Thereby, the text markers provide the context for a reliable attachment, which otherwise is highly ambiguous. The text markers can either surround the whole phrase as in (2.28a) or only parts of the phrase as in (2.28b).

(2.28) a. \( [_{NP} \text{“Einladung} \ [_{PP} \text{zur Enthauptung } ] ” ] \)
   invitation to the decapitation

b. \( [_{NP} \text{die schlagfertige “Frau} \ [_{PP} \text{mit den Hüten } ] ” ] \)
   the quick-witted woman with the hats

In any case, however, the text markers have to surround at least the PP and the head noun.

The number of iterations of the second level are, in principle, not limited. Three iterations, however, seem enough to cover the depth of the hierarchy present in the language data. The fact that not only the output of previous iterations but also of those phrasal categories built earlier in the same iteration are available for the current rules makes it possible to advance faster in the hierarchical structure building process.

The fact that the results of the rules are post-processed by Perl scripts allows to overgeneralize and underspecify parts of the grammar. Constraints and filters can be applied to the results, annotating only those structures which fulfill the constraints or pass the filters. Complex APs embedding PP or NP structures, e.g., can be restricted to contexts where they are included in a larger NP structure with at least one other element preceding the AP inside the NP as in (2.29).
2.4 General framework of YAC

(2.29) \[ NP \; die \; AP \; [PP \; von \; der \; Gemahlin] \; PP \; gegen \; das \; Fußpilzrisiko \]  
the of the wife against the athlete’s foot  
\[ AC \; empfohlenen \; ] \; Socken \]  
recommended socks

‘the socks recommended by the wife against the athlete’s foot’

Whether the constraint is fulfilled can be checked by the Perl-script after the actual rule has been processed.

Third Level

The third level can be seen as a finalizing level. It fulfills several purposes:

• chunkes of related but different categories can be subsumed under one category
• the category of certain chunks is changed
• coordination of maximal chunks
• decisions are made which need full recursive chunks

Category names of chunks which in fact belong to the same category are subsumed under one categorical name. The single structures have different names during the parsing process because of practical reasons. In the case of NPs, e.g., a difference is made between:

• NPs with determiner, which have the category \text{n}\text{p}
• NPs without determiner, which have the category \text{n}\text{cc}
• and base noun chunks, which have the category \text{n}\text{c}

During the parsing process, the difference is important. NPs without determiner can still be extended in pre-head position, while NPs with determiner cannot be extended to the left. Besides, if the noun chunks are embedded in larger constructions, NPs without determiner are usually more restricted than NPs with determiner. This is due to the fact that the morpho-syntactic information of NPs with determiner is usually less ambiguous. The base noun chunks are important to be able to trace certain structures back to their kernel, i.e., not to lose the kernel information during the parsing process. After the parsing process has been terminated, there is no reason to keep the different names for the three structures. On the contrary, the different names would make the resulting annotation less transparent for further applications.

Certain decisions require recursive chunk information, or are made more reliable by the chunk information. Adverbial and predicative adjectives share
Figure 2.9: TIGERSearch tree diagram of (2.29)
2.4 General framework of YAC

the same PoS-tag, and can only be differentiated by their actual usage. Consequent-ly, whether an AP is adverbially or predicatively used can only be determined by the usage of the AP. The usage of the AP can, however, only be determined when the chunking process has been terminated. In this case, if the AP is used as a modifier, i.e., if it is embedded in a larger structure, the AP is adverbially used, whereas, if it is not embedded, it is predicatively used. Headless NPs are another example. In this case, the NP is identified by a stranded NP followed by an AP, which is not embedded. In order to do so, however, the other chunks have to be already built.

The category of certain chunks can also only be determined after the main parsing process has been terminated. Chunks with adverbially used adjectives, e.g., are treated as APs during the parsing process. However, linguistically, they are really AdvPs. As has been argued before, the adverbial character of these chunks can only be determined after (recursive) chunk information is available. Thus, the true category of these chunks as AdvPs can no sooner be determined, than after the main parsing process.

Clause Level

In the clause level, the annotated phrasal structures are assembled into clausal structures. The rules are kept simple including any possible element of the respective clause. As the clause level runs only once, recursion is limited to recursive embedding of relative clauses, and simple appositions in finite, and infinite clauses.

Hierarchy Building

As said before, CQP does not allow overlapping structures of the same category. Therefore, the complete hierarchy of annotated structures has to be built after the parsing process itself. In order to do so, resulting structures of all parsing stages are collected throughout the process, and are stored in XML files. After the parsing process is completed the collected structures are combined into a hierarchy. As all intermediate structures are collected, there can be more than one analyses of the same phrase. Only the analysis with the largest span of words is taken to be included in the hierarchical structure. In order to determine whether a smaller phrasal structure is an embedded structure or only a smaller version of the same phrase, the head position is taken as reference. In other words, only the largest of overlapping phrasal structures with the same head position is included in the hierarchy, the smaller versions are neglected. Overlapping phrasal structures with different head positions, however, are all included in the hierarchy. This corresponds to the longest match strategy, which Abney used as well for the CASS grammar. The NPs in (2.30a–d) are overlapping NPs collected during the parsing process. The heads, representing the
head position, are marked in bold face. As can be observed, the NPs in (2.30a–c) have the same head. The NPs in (2.30a–b) are included in the NP in (2.30c), which is the largest of these three structures. Consequently, the NPs in (2.30a–b) are considered earlier versions of the NP in (2.30c). The NP in (2.30d) is also included in the NP in (2.30c), however, it does not share the same head. It is considered to be a different NP which is embedded in the larger NP. Thus, the NPs in (2.30c–d) are included in the hierarchy (2.30e), while the NPs in (2.30a–b) are neglected.

(2.30) a. \[ \text{[NP Faszination ]} \]
   fascination

   b. \[ \text{[NP gewisse Faszination des Schattens ]} \]
   certain fascination of the shadow

   c. \[ \text{[NP eine gewisse Faszination des Schattens ]} \]
   a certain fascination of the shadow

   d. \[ \text{[NP des Schattens ]} \]
   of the shadow

   e. \[ \text{[NP eine gewisse Faszination [NP des Schattens ]]} \]
   a certain fascination of the shadow

Whereas during the parsing process only the largest phrasal structure of the same category is available, the final version of annotated structures displays the full hierarchy of the recursive chunks. Thus, extraction processes have the full hierarchy available as knowledge source.
Chapter 3

Grammar Details

3.1 Introduction

The target groups of the following chapter are both:

- the potential users of corpora chunked by YAC, and
- grammar developers, who wish to develop similar grammars for other purposes, or languages, or who wish to carry over aspects of this grammar to another grammar type.

The first group is supposedly more interested in knowing what kind of information is annotated by YAC. The second group on the other hand needs detailed information about the grammar rules, what they include, and how they function. In order to serve both groups, and to make navigation through this chapter easy, I structured the chapter as follows. First I will give a brief survey of the annotated phrasal categories, and of the information annotated in association with them. Then I will give a very brief survey of how the rules are organized within the grammar of YAC. The main part of this chapter will then give detailed information about the annotation scheme, and the rules. Tables will provide an overview of annotation schemes, annotated features, and rules. Each phrasal category will be dealt with separately. The rules will be divided into first level, second level and third level rules. This should give the reader fast access to the desired information, and the possibility to skip information he is not interested in.

As described before in chapter 2, YAC does not annotate lexical categories itself. Tokenization and PoS-tagging, in this case, is performed by the TreeTagger, and lemma and agreement information is taken from the IMSLex. Taking this basic annotation as input, YAC annotates phrasal categories. As discussed before, within the framework of YAC two different groups of phrasal categories are differentiated:
• simple phrasal categories, and
• complex phrasal categories

It is sufficient to annotate simple phrasal categories at the first and possibly at the third level only. Complex phrasal categories are annotated at all three levels. During the second level the possible complex structure of the chunks is built as has been described in section 2.4.3 before. Within the framework of YAC the phrasal categories are divided as follows:

• simple phrasal categories
  – adverbial phrases (AdvP)
  – verbal complexes (VC)

• complex phrasal categories
  – adjectival phrases (AP)
  – noun phrases (NP)
  – prepositional phrases (PP)

Annotation scheme

Each phrasal category has its own annotation scheme, i.e., for each phrasal category different types of information are annotated. Only the information relevant for the respective category is annotated. As head lemma and lexical, semantic or structural properties are relevant for all categories, this information is available for all phrasal categories. Morpho-syntactic information, however, is annotated only for those phrasal categories which show agreement, and for which the information is necessary. Adverbs, e.g., do not show agreement, thus, morpho-syntactic information is not available. The head position is annotated only for complex phrasal categories. For simple phrasal categories it is not necessary as it is only used to build the hierarchy of structures. Some information is annotated only for one phrasal category. For APs, e.g., suffix information is annotated. By suffix information, in this case, we mean the last letter of the head lemma. The annotation scheme of each phrasal category will be listed in the subsequent sections. The different feature values can be single values, as well as multi-value feature sets. Single values are used, when the associated value is limited to a single item as is the case for head lemmas. Multi-value feature sets are used if the associated information can be ambiguous as, e.g., morpho-syntactic information, or can have more than one value, e.g., lexical-semantic properties.

The tables for the annotation scheme give lists the general feature values for each phrasal category, the CQP structure which includes the attributes of the values, and finally a short description of the feature value.
3.1 Introduction

Rule blocks

The grammar rules of YAC are organized in rule blocks, one rule block per phrasal category and annotation level. The rule blocks are annotated successively, each block in a separate CQP session. The results of each rule block are immediately available to following rule blocks. Consequently, the order of application is important as the rules of each block can build on the results of preceding blocks. The sequence of rule blocks should mirror the sequence of embedding. If the rule blocks are ordered correctly, we have to apply fewer iterations. This means at the same time that we save processing time.

Phrasal categories can take part in the embedding cycle actively or passively, or they do not take part in the embedding process at all. If they take part actively, they can be embedded in other structures and can embed other structures. If they take part passively, they can only be embedded in other structures, but cannot themselves embed other structures. If they do not take part in the embedding process, they are neither embedded by other structures, nor can they themselves embed other structures. Verbal complexes, e.g., do not take part in the embedding process, i.e., they neither embed other phrasal structures, nor are they embedded by other phrasal structures. AdvPs take part passively, i.e., they can be embedded by other chunks, but cannot themselves embed other structures. APs, NPs, and PPs can both take active and passive parts in embedding. APs, e.g., can be embedded in NPs, and are able to embed AdvPs, PPs and NPs. NPs can be embedded in PPs and APs, and themselves are able to embed APs, and PPs. PPs can be embedded by APs and NPs, and are able to embed NPs and AdvPs. It can be observed that the embedding process of the complex categories is circular. APs can be embedded in NPs, NPs in PPs, and NPs and PPs in APs, and so on. Figure 3.1 exemplifies this.

As the cycle of embedding indicates, we start by annotating AdvPs, as they can be embedded in other structures, but do not themselves embed other structures. We then enter the cycle with the AP rule block, and following the cycle going on to NPs, and finally PPs which terminate the first course of the cycle and the first level. Consequently, the second level starts annotating APs, moves on the NPs, and PPs. The second level moves through the cycle several times in order to build larger and larger structures iteratively. Although the number of runs is not limited in general, we run the second level three times.

In addition to the phrasal categories we annotate simple, mostly non-recursive, clauses. They are built combining the annotated phrasal categories to simple clause structures. Only relative clauses, and certain simple nominal appositions are embedded in other clauses. The purpose of these clauses is to ease extraction of linguistic and lexicographic information out of complex constructions.
Figure 3.1: Cycle of embedding
3.2 Simple phrasal categories

3.2.1 Adverbial Phrases (AdvP)

Adverbial phrases (AdvP) are phrasal structures with an adverb or adjective used adverbially as head. AdvPs - at least within the framework of YAC - are simple, mostly non-recursive structures which do not involve deep embedding. Most AdvPs consist of a single item - the head. A particle can function as pre-head modifier. In specific contexts, AdvPs can be extended to include other AdvPs as pre-head modifier.

Annotation scheme

An overview of the annotation scheme of AdvPs is given in Table 3.1.

<table>
<thead>
<tr>
<th>feature value</th>
<th>CQP</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>advp_f</td>
<td>lexical-semantic properties</td>
</tr>
<tr>
<td>h</td>
<td>advp_h</td>
<td>head lemma</td>
</tr>
</tbody>
</table>

Table 3.1: Annotation scheme of ADVP (adverbial phrase)

The annotation scheme of AdvPs comprises structural information, head lemma, and lexical-semantic properties.

Lexical-semantic properties

An overview of the lexical-semantic properties annotated as feature attributes for AdvPs is given in Table 3.2.

<table>
<thead>
<tr>
<th>values</th>
<th>description</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>adj</td>
<td>adverbially used adjective</td>
<td>3.1</td>
</tr>
<tr>
<td>dirfrom</td>
<td>directional source</td>
<td>3.2</td>
</tr>
<tr>
<td>dirto</td>
<td>directional path</td>
<td>3.3</td>
</tr>
<tr>
<td>loc</td>
<td>locative</td>
<td>3.4, 3.2, 3.3</td>
</tr>
<tr>
<td>norm</td>
<td>default</td>
<td></td>
</tr>
<tr>
<td>quot</td>
<td>enclosed in quotation marks</td>
<td>3.5</td>
</tr>
<tr>
<td>temp</td>
<td>temporal</td>
<td>3.6</td>
</tr>
</tbody>
</table>

Table 3.2: Lexical-semantic properties of ADVP (adverbial phrase)
Grammar Details

(3.1) schön  stark
       beautifully strong

(3.2) daher (from there); irgendwoher (from anywhere)

(3.3) heim (home); querfeldein (cross-country)

(3.4) innen (inside); überall (everywhere)

(3.5) “sehr bald” (very soon)

(3.6) jetzt (now); damals (at that time)

There are two major categories of lexical-semantic properties associated with AdvPs

- locative AdvPs (3.4, 3.3, 3.2)
  - marked with the feature attribute [loc]

- temporal AdvPs (3.6),
  - marked with the feature attribute [temp]

Locative AdvPs have two subclasses: directional path AdvPs marked with the feature attribute [dirto] as in (3.3), and directional source AdvPs marked with the feature attribute [dirfrom] as in (3.2). Locative AdvPs which fall into one of the subclasses have both the feature attribute of the subclass and the more general feature attribute loc. All other locative AdvPs only have the feature attribute loc as in (3.4). AdvPs enclosed in quotation marks are annotated with the feature attribute [quot] as in (3.5). AdvPs with an adverbially used adjective as head are marked with the feature attribute [adj] as in (3.1). All other AdvPs are annotated with the default feature attribute [norm].

First level grammar rules

An overview of the two first level grammar rules for AdvPs is given in Table 3.3.

<table>
<thead>
<tr>
<th>advp</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>advp</td>
<td>default AdvP</td>
<td>3.2, 3.3, 3.4, 3.6</td>
</tr>
<tr>
<td>advp_quot</td>
<td>AdvP in quotation marks</td>
<td>3.5</td>
</tr>
</tbody>
</table>

Table 3.3: First level ADVP rules

Only a small number of rules for AdvPs necessary because the structure of AdvPs within YAC is simple, and does not have many variations. Most AdvPs are covered by one general rule (advp), which is the default rule.
3.2 Simple phrasal categories

The rule states that an AdvP consists of an adverb as head and a possible particle as modifier in pre-head position.

The second rule (advp_quot) is an extension of the default rule.

It additionally allows adverb and adverbially used adjectives as pre-head modifiers. Surrounding quotation marks provide a reliable context for the additional attachment possibilities. They set clear borders to the range of the AdvP.

**Post-processing**  AdvPs with different lexical-semantic properties do not trigger different structures. Thus, it is not necessary to have a rule for each lexical-semantic property. The two rules mentioned above are sufficient to cover all structural variations. The lexical-semantic properties of AdvPs are determined during the post-processing of the rules by the Perl-scripts. Thereby, each head is compared with the different word lists available for adverbs. If the head is part of one of the lists, the respective lexical-semantic property is annotated as feature attribute.

**Third level grammar rules**

There is only one third level rules for AdvPs, see Table 3.4.

<table>
<thead>
<tr>
<th>advp</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>advp</td>
<td>adverbially used AP</td>
<td>3.1</td>
</tr>
</tbody>
</table>

Table 3.4: Third level ADVP rules
Grammar Details

The third level AdvP rule (advp) is not a structure building rule. It is only used to convert APs with adverbially used adjectives into AdvPs. We consider chunks with an adverbial adjective as head as AdvPs rather than as APs. Thus, we reinterpret and re-annotate these structures. The PoS-tag of adverbial and predicative adjectives is the same. The two functions cannot be differentiated without context information in the form of phrasal structures. Thus, the two functions are differentiated after the main parsing has been performed, and chunk information is available. The rule for AdvPs with adverbial adjectives as head searches for an AP with an adverbial or predicative adjective as head which is embedded in another phrasal structure, and modifies an adverb or another adjective. In this case, the predicative reading can be excluded in favor of the adverbial reading. The respective AdvP is marked by the feature [adj] to indicate its adjectival character.

3.2.2 Verbal Complexes (VC)

Verbal complexes are simple "non-recursive" chunks with almost no inner structure or hierarchy. VCs contain adjacent verbal elements which are not specified more closely. If the chunk contains a content verb, it is annotated as head of the chunk.

Annotation scheme

An overview of the annotation scheme of VC is given in Table 3.5.

<table>
<thead>
<tr>
<th>feature value</th>
<th>CQP</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>vc_f</td>
<td>lexical-semantic properties</td>
</tr>
<tr>
<td>h</td>
<td>vc_h</td>
<td>head lemma</td>
</tr>
</tbody>
</table>

Table 3.5: Annotation scheme of VC (verbal complexes)

The annotation scheme comprises the structure itself, the head lemma, as well as properties specifying the character of the chunk.

Lexical-semantic properties

An overview of the properties associated with VCs is given in Table 3.6.

(3.7) muss gerechnet werden
has counted to be
has to be counted
### Table 3.6: Lexical-semantic properties of VC (verbal complexes)

<table>
<thead>
<tr>
<th>values</th>
<th>description</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{in}f)</td>
<td>infinite</td>
<td>3.9</td>
</tr>
<tr>
<td>(\text{no}r)m</td>
<td>default</td>
<td>3.7</td>
</tr>
<tr>
<td>(\text{pre}f)</td>
<td>verb with separable prefix</td>
<td>3.8</td>
</tr>
<tr>
<td>(\text{zu})</td>
<td>infinite with ‘zu’</td>
<td>3.9</td>
</tr>
</tbody>
</table>

3.2 Simple phrasal categories

Most VCs are left underspecified with respect to their properties, and carry the default feature value \(\text{no}r\)m, an example is given in (3.7). Infinitival VCs with the infinitival marker \(\text{zu}\) (to) carry the feature values \(\text{in}f\) for infinitival and \(\text{zu}\), an example is given in (3.9). Other infinitival VCs are not marked as the tagging quality with respect to the difference between finite and infinitival verbal elements is not reliable enough. A differentiation would not provide important information as it would too often be false. Apart from infinitives with \(\text{zu}\), VCs which have a verb with a separable prefix as head are marked with a specific feature value, namely \(\text{pre}f\), an example is given in (3.8).

The VC itself comprises only the continuous verbal elements, and not the separable prefix. The head lemma of the VC, however, includes the separated prefix. In the example in (3.8) the VC includes only the verbal element \(\text{k}om\text{mt}\) (arrives), the head lemma of the VC, however, is \(\text{an}k\text{ommen}\) (to arrive). The separable prefix is added to the verb lemma after the main parsing process has been completed. At that stage, the prefix can be identified with good reliability as sentential constituents and clauses are already annotated. Possible separable prefixes can be:

- elements tagged as separable verbal prefix
- prepositions, and circumpositions not included in a PP

The prefix candidate has to be to the right of the verb. In between verb and prefix there can be an unlimited number of adjuncts, NPs, and complete subordinate clauses.
Grammar rules for verbal complexes (VC)

Because of the simple structure of VCs, VC rules operate in the first level only. An overview of the rules for VCs is given in Table 3.7.

<table>
<thead>
<tr>
<th>vc</th>
<th>rules</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>vc</td>
<td>default VC</td>
<td></td>
<td>3.7</td>
</tr>
<tr>
<td>vc_infzu</td>
<td>infinitival VC with 'zu'</td>
<td></td>
<td>3.9</td>
</tr>
</tbody>
</table>

Table 3.7: First level VC rules

Only two rules are necessary to annotate verbal complexes. The first rule \((vc)\) is the default rule. It states that the default VC consists of adjacent verbal elements which are not more closely specified. The second rule \((vc_{infzu})\) is a rule for verbal complexes with the infinitive marker \(zu\) (to). It is almost as simple as the default rule. It includes adjacent verbal elements and the infinitive marker \(zu\), which can occur at any position within the VC.

Single-item Verbs (V)

Apart from the VC structures, single verbal elements (V) are annotated. V structures are not phrasal structures in the narrow sense but rather single-item-units.

Annotation scheme An overview of the annotation scheme of these "single-item" verbs is given in Table 3.8

<table>
<thead>
<tr>
<th>v (verb)</th>
<th>feature value</th>
<th>CQP</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(f)</td>
<td>(v_f)</td>
<td>lexical-semantic properties</td>
</tr>
</tbody>
</table>

Table 3.8: Annotation scheme of V (verb)

The annotation scheme of VCs comprises structural information, as well as lexical-semantic properties of single-item verbs.

Lexical-semantic properties An overview of the properties associated with verbal elements is given in Table 3.9.

The features attributes differentiate between auxiliaries \((aux)\) as in (3.10), modal verbs \((mod)\) as in (3.12), and content verbs \((cont)\) as in (3.11).
3.3 Complex phrasal categories

### 3.3.1 Adjectival Phrases (AP)

Adjectival phrases (AP) are phrasal structures with attributive or predicative adjectives, as well as attributive indefinite pronouns with determiners as head.
Grammar Details

APs can comprise simple structures consisting of the adjectival head (3.13a) and possibly a simple modifier (3.13b), as well as complex structures involving complex (recursive) embedding (3.13c, Figure 3.2). The adjectival head in simple cases can be modified by particles or AdvPs (including AdvPs with adverbially used adjectives as head), for the more complex structures, the adjectival head takes PPs or NPs as modifiers or complements.

(3.13) a. schöne (beautiful)
   b. zu gutes too good
   c. [AP [PP aus [NP den Traditionen [NP der [AP legendären ]
      out of the traditions of the legendary Gewerkschaft [NP “Solidarität” ]] hervorgegangenen ]
      union “Solidarity” arising
      ‘arising out of the traditions of the legendary union “Solidarity” ’

Figure 3.2: TIGERSearch tree diagram of (3.13c)

Annotation scheme

An overview of the annotation scheme of APs is given in Table 3.11.

The annotation scheme comprises structural information, head lemma, and lexical-semantic properties. In addition to morpho-syntactic information, the
head position, and the “suffix”, or better the last letter, of the surface realization of the head is given. Head position and suffix information are available during the annotation process only. They are not part of the final annotation. As has been said before, the head position is necessary to determine the hierarchy of structures for the final annotation. The suffix information, i.e., the last letter of the adjectival head, is used to check agreement among embedded APs during the parsing process. The final annotation scheme comprises structural information, head lemma, lexical-semantic properties, and morpho-syntactic information.

### Lexical-semantic properties of APs

Depending on the head of the AC different lexical and semantic features are annotated. Most features are available for all AP structures, i.e., for base adjectival chunks as well as full AP structures, others are valid only for full AP structures. In other words, all properties valid for base adjectival chunks are valid for full AP structures, but not vice versa (feature inheritance). This has to do with the fact that some of the properties are triggered by the structure of APs.

**Lexical-semantic properties of base adjectival chunks (AC)** An overview of the lexical-semantic properties of ACs is given in Table 3.12.

Two of the annotated feature attributes are associated with the usage of the AC. Predicatively used ACs as in (3.14), e.g., are annotated with the feature pred, while attributive ACs as in (3.15) are annotated with the feature attr.

(3.14) [Es ist] klar.

It is clear

(3.15) klare [Sicht]

clear view

<table>
<thead>
<tr>
<th>feature value</th>
<th>CQP</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agr</td>
<td>ap_agr</td>
<td>agreement information</td>
</tr>
<tr>
<td>f</td>
<td>ap_f</td>
<td>lexical-semantic properties</td>
</tr>
<tr>
<td>h</td>
<td>ap_h</td>
<td>head lemma</td>
</tr>
<tr>
<td>hpos</td>
<td>ap_hpos</td>
<td>corpus position of head lemma</td>
</tr>
<tr>
<td>suff</td>
<td>ap_suff</td>
<td>last letter of head lemma</td>
</tr>
</tbody>
</table>

Table 3.11: Annotation scheme of AP (adjective phrase)
### Table 3.12: Lexical-semantic properties of AC (adjectival chunk)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>attr</code></td>
<td>attributive usage</td>
<td>3.15</td>
</tr>
<tr>
<td><code>brac</code></td>
<td>enclosed in brackets/parentheses</td>
<td>3.20</td>
</tr>
<tr>
<td><code>invar</code></td>
<td>invariant</td>
<td>3.18</td>
</tr>
<tr>
<td><code>norm</code></td>
<td>default</td>
<td>3.16</td>
</tr>
<tr>
<td><code>pidat</code></td>
<td>attributive indefinite pronoun with determiner</td>
<td>3.18</td>
</tr>
<tr>
<td><code>pred</code></td>
<td>predicative</td>
<td>3.14</td>
</tr>
<tr>
<td><code>quot</code></td>
<td>enclosed in quotation marks</td>
<td>3.19</td>
</tr>
<tr>
<td><code>vder</code></td>
<td>deverbal</td>
<td>3.17</td>
</tr>
</tbody>
</table>

Other feature attributes are associated with the characteristic of the head of the ACs. AC with an indefinite pronoun with determiner as head (3.16) are annotated with the feature `pidat`\(^1\). AC with a deverbal adjective as head (3.17) are associated with the feature `vder`, and AC with an invariant adjective as head, as in (3.18), are marked with the feature `invar`.

(3.16) viele Studenten

many students

(3.17) empfohlene

recommended

(3.18) lila (purple)

Similar to AdvPs ACs surrounded by textual markers are associated with the respective features. AC in quotation marks are annotated with the feature `quot` (3.19), and AC in brackets or parentheses are marked with the feature `brac` (3.20).

(3.19) “abenteuerlich”

adventurous

(3.20) (falschen)

false

**Lexical-semantic properties of APs**  An overview of the lexical-semantic properties of APs is given in Table 3.13.

Two of the features are related to the structure of the AP. The feature attribute `np` (3.21, Figure 3.3) indicates that the AP embeds an NP, while the feature attribute `pp` (3.22, Figure 3.4) indicates an embedded PP.

---

\(^1\)The name of the feature is taken from the corresponding PoS-tag of the STTS-tagset
### 3.3 Complex phrasal categories

<table>
<thead>
<tr>
<th>feature values for $f$</th>
<th>values</th>
<th>description</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>meas</td>
<td>meas</td>
<td>measure</td>
<td>3.23</td>
</tr>
<tr>
<td>np</td>
<td>np</td>
<td>embedded NP</td>
<td>3.21</td>
</tr>
<tr>
<td>pp</td>
<td>pp</td>
<td>embedded PP</td>
<td>Figures 3.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Figures 3.4</td>
</tr>
</tbody>
</table>

Table 3.13: Lexical-semantic properties of AP (adjective phrase)

(3.21) \[ AP \ [NP \text{Pferde}], [NP \\ Esel \\], [NP \text{Ziegen]} \text{und} [NP \text{Hunde} \text{umfassenden}] \]

comprising

comprising horses, donkeys, goats and dogs

(3.22) \[ AP \ [PP \text{von der Gemahlin}] [PP \text{gegen das Fußpilzrisiko}] \]

by the wife against the athlete's foot

recommended

‘recommended by the wife against athlete's foot’

Figure 3.3: TIGERSearch tree diagram of (3.21)

The third feature $\text{meas}$ (3.23) indicates the lexical property of the head as being a measure adjective ($\text{höherer}$, higher).

(3.23) rund 600 Millionen Mark höherer

around 600 million Mark higher
Grammar Details

First level AP rules

APs are annotated in all three levels. During the first level, simple non-recursive AP structures (base ACs) are annotated. An overview of the rules for base ACs is given in Table 3.14.

<table>
<thead>
<tr>
<th>rules</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ac</td>
<td>default AC</td>
<td>3.24, 3.25</td>
</tr>
<tr>
<td>ac_quot</td>
<td>AC in quotation marks</td>
<td>3.26, 3.27</td>
</tr>
<tr>
<td>ac_brac</td>
<td>AC in brackets</td>
<td>3.28, 3.29</td>
</tr>
<tr>
<td>(ac_coord)</td>
<td>coordination of ACs (used in np macros)</td>
<td></td>
</tr>
<tr>
<td>(ac_agrees)</td>
<td>agreeing ACs (used in ac macros)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.14: First level AP rules

(3.24) kleinen (small)

(3.25) zu blau
  too blue

(3.26) “sehr introvertiert”
  very introverted
3.3 Complex phrasal categories

(3.27) “abenteuerlich”
adventurous

(3.28) (falschen)
false

(3.29) (noch vergleichsweise milden)
still comparably mild

The default rules (ac) consist only of a single token, the adjectival head, and possibly a particle as in (3.24, 3.25).

MACRO ac(0)

(pos = "PTKA|PTKNEG"]?
[@[pos = "ADJ.*|PIDAT"]]

In addition this default rule, there are two additional rules for APs in quotation marks (ac_quot) as in (3.26, 3.27) and APs in brackets or parentheses (ac_brac) as in (3.28, 3.29).

MACRO ac_quot(0)

/quota[]
 [pos = "PTKA|PTKNEG"]?
 [pos = "ADJD|ADV"]* 
[@[pos = "ADJ.*|PIDAT"]
 /quote[]

These rules include besides the adjectival head and possible particles optional pre-head adverbial modifiers. The adverbial modifiers can either be adverbs or adverbia! ally used adjectives. The grammar code of the rule ac_brac is almost identical to the grammar code of ac_quot, only the surrounding text markers are different, quotation marks for ac_quot, and parentheses or brackets for ac_brac.

The rules in Table 3.14 enclosed in parentheses are rules not directly applied to the corpus but are used in other rules. The first of these rules ac_coord is used in NP rules to include optional APs with optional coordination.

MACRO ac_coord(0)


Grammar Details

<ac>
  a: [._ac_f contains "attr"]
  []*
</ac>

It comprises an attributive AC followed by an unspecified number of agreeing ACs, i.e., sharing the same suffix. The agreeing ACs are represented by the secondary rule ac_agrees.

MACRO ac_agrees($0=AcRef)
(
  <ac>
    [ (_._ac_f not contains "invar" & _.ac_f contains "attr")
      -> (_._ac_suff = $0.ac_suff) ]
    []*
  </ac>
)

It states that a base adjectival chunks (AC) which is not invariant (not contains invar) and is used attributively (contains attr) has to have the same suffix as another AC which is referred to by the variable $0. The label a which is set on the first AC in the rule ac_coord, is handed over to the rule ac_agrees to make a comparison of the suffixes possible. The agreeing ACs can be coordinated optionally using coordinate conjunctions or commas.

Second level rules

In the second level, more complex APs are annotated, including recursive constructions. An overview of the rules is given in Table 3.15.

The default rule (ap) annotates relatively simple APs consisting of an adjectival chunk (AC) (annotated in the first level) as kernel and optionally adverbial phrases as modifiers (3.30, 3.31).

(3.30) meist nicht gerade "bürgerlichen"
mostly not really civil

(3.31) finanziell nicht gerade reich gesegnete
financially not really richly blessed

80
### 3.3 Complex phrasal categories

<table>
<thead>
<tr>
<th>ap</th>
<th>rules</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ap_mod</td>
<td>AP with adverbial modifier</td>
<td>3.30, 3.31</td>
<td></td>
</tr>
<tr>
<td>ap</td>
<td>default AP</td>
<td>3.30, 3.31</td>
<td></td>
</tr>
<tr>
<td>ap_pp</td>
<td>complex AP embedding PPs</td>
<td>3.40, 3.41 Figures 3.5, 3.6,</td>
<td></td>
</tr>
<tr>
<td>ap_np_first</td>
<td>complex AP embedding NPs (first run of second level)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ap_np</td>
<td>complex AP embedding NPs</td>
<td>3.42, 3.43 Figures 3.7, 3.8,</td>
<td></td>
</tr>
<tr>
<td>ap_quot</td>
<td>AP in quotation marks</td>
<td>3.32, 3.33</td>
<td></td>
</tr>
<tr>
<td>ap_meas_first</td>
<td>AP with measure adjective as head (first run of second level)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ap_meas</td>
<td>AP with measure adjective as head</td>
<td>3.34, 3.35</td>
<td></td>
</tr>
<tr>
<td>(ap_coord)</td>
<td>AP coordination (used in ncc macros)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ap_coord)</td>
<td>AP coordination with label (used in ncc macros)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ap_coord_2)</td>
<td>coordinated APs with label</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ap_agrees)</td>
<td>agreeing APs (used in ap macros)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(advp_mod)</td>
<td>adverbial modifier of AP (used in ap macros)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ap_secure)</td>
<td>secure context for complex AP (used in ap macros)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.15: Second level AP rules
The grammar code includes two other AP rules `ap_secure` and `ap_mod`. The rule `ap_mod` is responsible for the adverbial modifiers.

```plaintext
MACRO ap_mod(0)
(
    /ap_secure[]
    /advp_mod[]+
    <ac>
        @[]
        []*
    </ac>
)
;
;

Adverbial modifiers can be either an AdvP or an adverbially used AC. The attachment of these adverbial modifiers is subject to ambiguity. Thus, adver-
bial modifiers are restricted to certain reliable contexts. The secondary rule `ap_secure` guarantees a reliable context.

MACRO ap_secure(0)
(
    # cardinal number
    [pos = "CARD"]
    |
    # AC
    [ac & _.ac_f not contains "invar" & _.ac_f contains "attr"]
    [word = "," | pos = "KON"]?
    |
    # AP
    [ap & _.ap_f not contains "invar" & _.ap_f contains "attr"]
    [word = "," | pos = "KON"]?
    |
    # determiner
    [pos = "ART|PDAT|PIAT|PIDAT|PPOSAT|PRELAT|PWT" & !np]
    |
```

82
The reliable context can be either a cardinal, an AC/AP which is not invariant, a
determiner, or a preposition which is not yet part of a PP If the reliable context
is an AC or AP, there can be a comma or a coordinating conjunction in between.

Similar to the AC rules of the first level, there is a rule for APs in quotation
marks (`quot()`), which allows to extend the default rule by optional adverbial
modifiers in pre-head position and surrounding quotation marks (3.32, 3.33).

(3.32) “ nicht besonders komfortabel ”
not very comfortable

(3.33) “ schwer psychisch krank ”
heavily psychologically ill

In this case, the quotation marks make an extra reliable context unnecessary.

MACRO ap_quot(0)
( 
  /quota[]
  /advp_mod[]*
  /region(ac, '@')
  /quote[]
)
);

The third AP rule in the second level is the rule for APs with measure adjectives as heads (`ap_meas`).

MACRO ap_meas(0)
( 
  /cardcoord[]?
  <ncc>[_._ncc_f contains "meas"][]*/ncc>
  @[lemma = RE($adj_Measure) & pos = "ADJA"]
)
);

In this case, the adjective heads are modified by noun chunks (NCs) with mea-
sure nouns as head in pre-head position as in (3.34).

(3.34) Meter hohen
meter-high
Optionally, cardinal numbers or NC with cardinal nouns as head can precede the measure NC as in (3.35, 3.36).

(3.35) rund 600 Millionen Mark höherer
around 600 million mark higher

(3.36) 5,4 Meter lange
5.4 meters-long

The cardinal elements can optionally be coordinated by comma, coordinating conjunction. The position of the coordinating conjunction can also be filled by prepositions, attributive possessive pronouns, and articles.

```plaintext
MACRO cardcoord(0)
(
    /card_elements[]
    (
        /comma[]
        /card_elements[]
    )*
    # "coordinating" elements
    [pos = "KON|APP.*|ART|PPOSAT"]
    [pos = "AD.*"]?
    /card_elements[]
)
)
;

(3.37) 4 bis 5 Meter hohe
4 to 5 meters-high

(3.38) 4 meiner 5 Schwestern
4 of my 5 sisters

(3.39) 4 der 5 Kinder
4 of the 5 children
```

Depending on the parsing stage, there are two similar ap_meas-rules. During the first run of the second level, the measure NC corresponds to the base noun chunks annotated in the first level. During later runs, the measure NC can correspond to more complex noun chunks (ncc) annotated during the second level only. These ncc structures are noun chunks without determiner which are used during the chunking process only².

More complex are the AP rules embedding PP (ap_pp) as in (3.40, 3.41, Figures 3.5, 3.6) or NP structures (ap_np) as in (3.42, 3.43, Figures 3.7, 3.8).

²See section 3.3.2 for more details
3.3 Complex phrasal categories

(3.40) \[ AP \ [PP \ auf \ der \ natürlichen \ Sprache] \ basierendes] \]
    on the natural language based

‘based on natural language’

(3.41) \[ AP \ [PP \ für \ Kinder] \ [PP \ im \ Sommer] \ sicher \ sehr \ attraktive] \]
    for children in the summer certainly very attractive

‘during the summer certainly very attractive for children’

Figure 3.5: TIGERSearch tree diagram of (3.40)

The AP rule ap_pp comprises an AC as kernel of the AP, which is modified by PPs in pre-head position.

MACRO ap_pp(0)
{
    /ap_secure[]
    # optional adverbial modifiers
    /advp_mod[]*
    # one or more embedded pp s
    /cpp[]+
    # optional adverbial modifiers
    /advp_mod[]*
    # head: attributive AC
The number of PPs is not specified. There can be a single PP as in (3.40, Figure 3.5), as well as several PPs which can either be coordinated or simply adjacent to one another as in (3.41, Figure 3.6). The embedded PPs are represented by the secondary PP rules (see pp. 129). Adverbial modifiers can precede the PP modifiers, but they can also hold the position between the PP modifiers and the AC.

APs embedding NPs are constructed similarly. The kernel is an AC, which embeds NPs in pre-head position. Optionally there can be an adverbial modifiers in front of the embedded NPs.

```
MACRO ap_np(0)
(
    /ap_secure[]
    # optional modifiers:
    /advp_mod[]*
    @[pos = "APP.*" & !pp]?
    # one or more embedded nps
    (/region[np] [/region[ncc]])+
```

86
3.3 Complex phrasal categories

The embedded NP structures can be either of the type np (noun chunks with determiner) or of the type ncc (noun chunks without determiner). During the first run of the second level ncc structures are not available, thus, there is a variant of the rule ap_np for the first level (ap_np_first). The rule is almost identical, but instead of ncc structures nc structures are embedded.

As for the PPs in the rule ap_pp, the number of the embedded NPs is not specified. There can be a single NP as in (3.42), as well as several NPs which can optionally be coordinated as in (3.43).

(3.42) \[ _{AP} \[ _{NP} \text{der “Inkatha-Partei” } \] angehörenden] to the “Inkatha-Partei” belonging

‘belonging to the “Inkatha-Party”’

(3.43) \[ _{AP} \[ _{NP} \text{Pferde}, _{NP} \text{Esel}, _{NP} \text{Ziegen} und _{NP} \text{Hunde} \] horses, donkeys, goats and dogs

umfassenden] comprising

‘comprising horses, donkeys, goats and dogs’

The optionally coordinated NPs are represented by the secondary NP rule np_coord (np_coord_first in the case of the ap_np_first), which is described in section 3.3.2.

The embedding of NPs and PPs is subject to ambiguity, and noise. Thus, it requires a reliable context. The reliable context is the same as for the attachment of adverbial modifiers in the default AP rule ap. It is represented by the secondary rule ap_secure which has already been described above, repeated here for convenience as being any of the following:

- a cardinal element,
- an attributive AP belonging to the same NP,
- a determiner, or
- a preposition which is not yet part of a PP
Grammar Details

Figure 3.7: TIGERSearch tree diagram of (3.42)

Figure 3.8: TIGERSearch tree diagram of (3.43)
The rules in Table 3.14 enclosed in parentheses are rules which are not directly applied to the corpus but are used in other rules. The rules $\text{ap\_mod}$ and $\text{ap\_secure}$ have already been described above. The other two secondary AP rules in the second level are $\text{ap\_coord}$, $\text{ap\_coord\_2}$ and $\text{ap\_agrees}$.

Similar to the first level rule $\text{ac\_coord}$ the second level rule $\text{ap\_coord}$ is used for coordinated APs embedded in NP structures. The only difference is that $\text{ap\_coord}$ coordinates AP structures, whereas $\text{ac\_coord}$ coordinates base adjectival chunks (AC). It includes a number of adjacent APs, which are optionally coordinated by a coordinate conjunction or a comma, and which have to agree with each other.

\begin{verbatim}
MACRO ap_coord($O=ApRef)
(
  <ap>
    $O: [_.ap_f contains "attr"]
    []*
  </ap>
  /ap_coord_2($O)*
)
;

The agreement is handled by labels, which allow comparison of different positions within the rule. There are two variants of this rule with the same name (ap_coord). In the first, the label is set within the rule itself. In the second, the label is handed over from outside the rule, as is displayed in the grammar code above.

The coordinated APs are represented by the rule $\text{ap\_coord\_2}$.

\begin{verbatim}
MACRO ap_coord_2($O=ApRef)
(
  [pos = "CON" | word = "," ]?
  /ap_agrees($O)
)
;

It comprises an optional coordinated conjunction or comma and the secondary rule $\text{ap\_agrees}$, which represents attributive APs which agree with another AP

\begin{verbatim}
MACRO ap_agrees($O=ApRef)
(
  <ap>
  [ (_.ap_f not contains "invar" & _.ap_f contains "attr")
     -> (_ap_suff = $O.ap_suff) ]
  []*
  </ap>

89
\end{verbatim}
Grammar Details

The rule `ap_agrees` is similar to the first level rule `ac_agrees`. It comprises an attributive AP the head of which is not invariant, and which agrees with another AP which is related by a variable referring to a label. The difference to the first level rule `ac_agrees` is that the AP structures are compared and not AC structures.

Third level AP rules

Third level AP rules are mainly used to differentiate between predicatively and APs used attributively. The following examples show the lemma schön (beautiful) with different usages.

(3.44) \[ NP \text{ eine } [_{AP} \text{ schöne } ] \text{ Frau } \]
\[ a \text{ beautiful woman} \]

(3.45) \[ NP \text{ die Frau } \text{ ist } [_{AP} \text{ schön } ] \]
\[ \text{The woman is beautiful} \]

(3.46) \[ NP \text{ eine } [_{AP} [_{AdoP} \text{ schön } ] \text{ starke } ] \text{ Frau } \]
\[ a \text{ beautifully powerfull woman} \]

In (3.44) the adjective is used attributively, in (3.45) it is used predicatively, and in (3.46) it is adverbially used. The different usages can only be differentiated with respect to chunk information, or inflection. The adjective in (3.44) is embedded in an NP and agrees with the head noun. It modifies the noun, and, thus, is used attributively. The adjective in (3.45) is not embedded in a NP and does not agree with the noun but rather with the verb. It is used predicatively. The adjective in (3.46) is embedded in an NP and does not show inflection in the form of agreement information as it is used adverbially. The tagger already differentiates between the inflected attributive adjective in (3.44) and the adverbial and predicatively used adjectives in (3.46) and (3.45). Adverbial and predicative adjectives are difficult to differentiate on token level, however, if chunk information is available, it is easy to tell them apart. The adjective is used adverbially, if it is embedded in an NP, and predicatively used if it is not.

Adverbially used APs are already re-interpreted by a third level AdvP rule as has been presented in section 3.2.1. As the difference between attributive and predicatively used adjectives is distributional, only two very simple rules are needed. An overview of third level AP rules is given in Table 3.16.

The first rule `ap_attr` comprises attributive APs.
3.3 Complex phrasal categories

<table>
<thead>
<tr>
<th>rules</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ap_pred</td>
<td>predicatively used AP</td>
<td>3.1</td>
</tr>
<tr>
<td>ap_attr</td>
<td>predicatively used AP</td>
<td>3.1</td>
</tr>
</tbody>
</table>

Table 3.16: Third level AP rules

MACRO ap_attr(0)
{
    <ap>
        [_.ap_f contains "attr"]
        []*
    </ap>
}
;

The second rule ap_pred comprised predicative APs.

MACRO ap_pred(0)
{
    <ap>
        [_.ap_f contains "pred"
            & !np]
        []*
    </ap>
}
;

In order to differentiate them from adverbially used adjectives, the APs are required to be inside a NP structure. In addition to these two rules all base adjectival chunks are collected.

3.3.2 Noun Phrases (NP)

Noun phrases (NP) are phrasal structures with nouns as head. Foreign language material is also considered to be NP material as it usually stands for nominal entities or fixed expressions. NPs can comprise simple structures consisting of a single noun (3.47a) or pronoun with optional determiners (3.47b) or cardinals (3.47c), as well as complex structures embedding complex APs (3.47d, Figure 3.9), and recursively embedding other NPs (3.47e, Figure 3.10). NPs, however, can also be headless, i.e., the actual head of the phrase is missing (3.47f).

(3.47) a. Oktober (October)
Grammar Details

b. der Mann (the man)
c. drei Meter (three meters)
d. \[NP \text{die acht } [AP [PP aus } [NP \text{den Traditionen } [NP \text{der } [AP \\
the eight out of the traditions of the legendären } ] \text{Gewerkschaft } [NP \text{“Solidarität” } ]])
legendary union “Solidarity” hervorgegangenen ] Parteien ] arising parties
‘the eight parties arising out of the traditions of the legendary union “Solidarity”’
e. \[NP \text{die } [AP \text{percussive ] Version } [NP \text{des } [AP [PP zu } [NP \text{Tode } \\
the percussive version of the to death genudelten ] Bolero ]]
noodled Bolero
‘the percussive version of the bolero played to death’
f. \[NP \text{die } [AP \text{jüngste } ] \text{PRO } \\
the youngest PRO

Annotation scheme

An overview of the annotation scheme of NPs is given in Table 3.17.

<table>
<thead>
<tr>
<th>feature value</th>
<th>CQP</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agr</td>
<td>np_agr</td>
<td>agreement information</td>
</tr>
<tr>
<td>f</td>
<td>np_f</td>
<td>lexical-semantic properties</td>
</tr>
<tr>
<td>h</td>
<td>np_h</td>
<td>head lemma</td>
</tr>
<tr>
<td>hpos</td>
<td>np_hpos</td>
<td>corpus position of head lemma</td>
</tr>
</tbody>
</table>

Table 3.17: Annotation scheme of NP (noun phrase)

The annotation scheme of NPs comprises structural information, head lemma, and lexical-semantic properties. Additionally, the head position is available during the parsing process in order to determine the hierarchy of NP structures for the final annotation.

Lexical-semantic properties

A variety of lexical-semantic properties are annotated for NP structures. Contrary to the lexical-semantic properties annotated for APs, most of the lexical-semantic properties annotated for NPs can trigger differences within the local
3.3 Complex phrasal categories

Die acht aus den Traditionen der legendären Gewerkschaft Solidarität hervorgegangenen Parteien.

Figure 3.9: TIGERSearch tree diagram of (3.47d)
Figure 3.10: TIGERSearch tree diagram of (3.47e)
structure of the NPs. In other words, they can lead to a specific structure of the base noun chunk (NC). The lexical-semantic property itself, however, is mostly determined by the head lemma.

**Lexical-semantic properties of base noun chunks (NC)** An overview of the lexical-semantic properties annotated as feature attributes for base noun chunks (NC) is given in Table 3.19.

<table>
<thead>
<tr>
<th>values</th>
<th>description</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>brac</td>
<td>enclosed in brackets/parentheses</td>
<td>3.48</td>
</tr>
<tr>
<td>card</td>
<td>cardinal structure</td>
<td>3.56</td>
</tr>
<tr>
<td>date</td>
<td>date</td>
<td>3.51</td>
</tr>
<tr>
<td>loc</td>
<td>locative</td>
<td></td>
</tr>
<tr>
<td>meas</td>
<td>measure</td>
<td>3.57</td>
</tr>
<tr>
<td>ne</td>
<td>named entity</td>
<td>3.53</td>
</tr>
<tr>
<td>news</td>
<td>news agency marker</td>
<td>3.55</td>
</tr>
<tr>
<td>norm</td>
<td>default</td>
<td></td>
</tr>
<tr>
<td>numb</td>
<td>list item</td>
<td>3.58</td>
</tr>
<tr>
<td>quot</td>
<td>enclosed in quotation marks</td>
<td>3.49</td>
</tr>
<tr>
<td>sport</td>
<td>sports club</td>
<td>3.54</td>
</tr>
<tr>
<td>street</td>
<td>street address</td>
<td>3.59</td>
</tr>
<tr>
<td>tel</td>
<td>telephone number</td>
<td>3.60</td>
</tr>
<tr>
<td>temp</td>
<td>temporal</td>
<td>3.50</td>
</tr>
<tr>
<td>year</td>
<td>year</td>
<td>3.52</td>
</tr>
</tbody>
</table>

Table 3.18: Lexical-semantic properties of NC (noun chunk)

Similar to APs, NCs enclosed in parentheses (3.48) and quotation marks (3.49) are marked with the features `brac` and `quot` respectively.

(3.48) ( SPD )

(3.49) “Kommerzielle Koordinierung”

commercial coordination

Three of the annotated features are associated with temporal aspect. NCs with temporal nouns as head are marked by the feature `temp` (3.50). Dates are marked by the feature `date` (3.51), and year dates are marked by the feature `year` (3.52).

(3.50) Ende Mai

end may
end of May
Grammar Details

(3.52) 1984

The latter two are also marked by the feature temp, indicating their temporal aspect.

Other NC features are associated with named entities. Proper nouns, and multi-word proper nouns are marked by the feature ne (3.53).

(3.53) Osteuropa

NC representing sport clubs are marked by the feature sport (3.54).

(3.54) FC Hansa Rostock

News agencies headers are marked by the feature news (3.55)

(3.55) LONDON ( afp / rtr / AP )

Three properties are associated with numeric aspects. NCs with cardinal nouns as head are marked by the feature card (3.56).

(3.56) fast 480 Millionen
almost 480 million

NCs with a measure noun as head are marked by the feature meas (3.57).

(3.57) fünf Prozent
five percent

NCs, where the head lemma can be modified by a cardinal in post-head position are marked by the feature numb (3.58)

(3.58) Paragraph 65
paragraph 65

The following properties are associated with a more or less locative aspect. NC representing street addresses are marked by the feature street (3.59), and additionally by the feature loc.

(3.59) Museumsgasse 1

NC representing telephone numbers are marked by the feature tel (3.60).

(3.60) Tel. 49 50_999
phone 49 50_999

NC including truncated nouns are marked by the feature trunc (3.61).

(3.61) Verantwortungs-, Solidariäts-, Gerechtigkeits-, Gleichheits- und
responsibility, solidarity, justice, equality and
Freiheitsdenken
liberty thinking
3.3 Complex phrasal categories

Lexical-semantic properties of NPs  The lexical-semantic properties of base NC are projected to the NP structures. In addition to the lexical-semantic properties introduced on base noun chunk level, the lexical-semantic properties given in Table 3.19 are annotated at the NP level.

<table>
<thead>
<tr>
<th>values</th>
<th>description</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>poss</td>
<td>possessive NP modifier replacing determiner</td>
<td>3.62</td>
</tr>
<tr>
<td>pron</td>
<td>pronominal</td>
<td>3.66</td>
</tr>
<tr>
<td>refl</td>
<td>reflexive</td>
<td>3.67</td>
</tr>
<tr>
<td>rel</td>
<td>relative</td>
<td>3.63, 3.69</td>
</tr>
<tr>
<td>wh</td>
<td>wh-phrase</td>
<td>3.65, 3.68</td>
</tr>
</tbody>
</table>

Table 3.19: Lexical-semantic properties of NP (noun phrase)

Most of the lexical-semantic properties annotated at NP level specify the character of the determiner. NPs, where the determiner is replaced by a prenominal genitive modifier functioning as possessive marker are marked by the feature \textit{poss} (3.62).

(3.62) \[
\left[ N_P \right. \left[ N_P \text{Peters} \right] \text{Fahrrad} \]
Peter’s bicycle

NPs with an attributive relative pronoun as determiner are marked by the feature \textit{rel} (3.63), while NPs with an attributive interrogative pronoun are marked by the feature \textit{wh} (3.65).

(3.63) \[
\left[ \text{der Hund,} \right. \left. \text{der} \right]
the dog which

(3.64) \[
\left[ \text{die Frau,} \right. \left. \text{deren Kunst} \right]
the women, whose art

(3.65) \[
\text{welche Frage}
which question

Pronominal NPs are marked by the feature \textit{pron} (3.66).

(3.66) \[
\text{dieser (this)}
Reflexive pronouns are additionally marked by the feature \textit{refl} (3.67).
Grammar Details

(3.67) sich (itself)

Interrogative pronouns are additionally marked by the feature \( \text{wh} \) (3.68)

(3.68) welcher (which)

Substitutive relative pronouns are additionally marked by the feature \( \text{rel} \) (3.69).

(3.69) [die Frau] deren Kunst

the women, whose art

First level grammar rules

NPs, as well as APs, are annotated at all three levels. Thereby, one has to
differentiate between base noun chunks without determiner (NC), and base
noun chunks with determiner or pronominal NPs (NP). An overview of the first
level grammar rules for NCs is given in Table 3.20

<table>
<thead>
<tr>
<th>nc</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>nc_ne</td>
<td>named entities</td>
<td>3.53, 3.80, 3.81, 3.82</td>
</tr>
<tr>
<td>nc_news</td>
<td>news agency marker</td>
<td>3.55, 3.84</td>
</tr>
<tr>
<td>nc_quot</td>
<td>NC in quotation marks</td>
<td>3.49, 3.71</td>
</tr>
<tr>
<td>nc_brac</td>
<td>NC in brackets</td>
<td>3.48, 3.70</td>
</tr>
<tr>
<td>nc_trunc</td>
<td>NC including truncated elements</td>
<td>3.61</td>
</tr>
<tr>
<td>nc_temp</td>
<td>NC with temporal head</td>
<td>3.50, 3.73</td>
</tr>
<tr>
<td>nc_temp_brac</td>
<td>NC with temporal head in parentheses</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(brackets)</td>
<td></td>
</tr>
<tr>
<td>nc_year</td>
<td>year</td>
<td>3.52, 3.72</td>
</tr>
<tr>
<td>nc_date</td>
<td>date</td>
<td>3.51, 3.75, 3.76</td>
</tr>
<tr>
<td>nc_card</td>
<td>NC with cardinal noun as head</td>
<td>3.56, 3.85</td>
</tr>
<tr>
<td>nc_meas</td>
<td>NC with measure noun as head</td>
<td>3.57</td>
</tr>
<tr>
<td>nc_numb</td>
<td>numbered NC</td>
<td>3.58</td>
</tr>
<tr>
<td>nc_street</td>
<td>street address</td>
<td>3.59</td>
</tr>
<tr>
<td>nc_tel</td>
<td>NC specifying a telephone number</td>
<td>3.60, 3.87</td>
</tr>
<tr>
<td>nc_sport</td>
<td>sport clubs</td>
<td>3.54, 3.83</td>
</tr>
<tr>
<td>nc_rest</td>
<td>default NC</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.20: First level NC rules

Similar to APs, there are base noun chunk rules for structures enclosed in
brackets and parentheses. Below the code for the rule nc_quot is given.
Both brackets, parentheses and quotation marks provide a reliable context. Thus, adjacent nouns, proper nouns (3.70) and foreign language material (3.71) can be combined to a phrasal structure. It can be reliably assumed that they belong together. The nouns can also optionally be coordinated.

(3.70) ( Heiliger Islamischer Krieg )

Holy Islamic War

(3.71) “ persona non grata ”

There are several rules used for annotating NCs with a temporal aspect. The first rule nc_year searches for year dates.

MACRO nc_year(0)
( # either four cardinals beginning with 1 or 2
  # 1973
  [word = "[12][0-9]{3}"]
  |
  # or two cardinals preceded by a '
  # '73
  [word = ":"]
  [word = "[0-9]{2}"]
)
;

It searches for tokens consisting either of four cardinal numbers beginning with 1 or 2 (3.52) or of two numbers which are preceded by an apostrophe (3.72).

(3.72) ’73

The second rule nc_temp searches for temporal NCs.
The head of the structure is a temporal noun such as *Jahr* (year) or *März* (march). The temporal noun can be preceded by a pre-temporal noun such as *Ende* (end), *Mitte* (middle) (3.50). After the head, there may be a post-head modifier, which can either be a post-temporal adverb such as *früher* (earlier) (3.73) or a year cardinal (3.74).

(3.73) Wochen später

week later

(3.74) Jahr 1984 (year 1984)

The year cardinal is represented by the rule `nc_year`. The third rule `nc_temp_brac` is similar to the rule `nc_temp`.

MACRO nc_temp_brac(0)
(  /braca[]
  # temporal noun chunk
  /nc_temp[]
  (    [pos = "APP.*|KON" | word = ","]
    # temporal noun chunk
    /nc_temp[]
  )*
  /brace[]
)

;
The only difference is that the structure can optionally be coordinated and is enclosed in parentheses.

The fourth rule for temporal noun chunks is \texttt{nc\_date}.

\begin{verbatim}
MACRO nc\_date(0)
  (
    # optional weekday introduction
    # Montag den, (Monday the,)
    (  
      [lemma = RE($\text{name\_weekday}$)]
      [word = ","]
      [lemma = "d"]?
    )?
    # optional coordination of day and month
    (  
      # one token: day.month
      # 22. or 22.7. or 22.07.
      [word = "[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}/"]
      |
      # two tokens: day. month
      # 22. Juli (July 22nd)
      [word = "[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}/" ]?
      [lemma = RE($\text{name\_month}$)]
    )  
    [pos = "KON" | word = "," | lemma = "bis"]
  )?
  # date
  (  
    # one token: day.month.year
    # 22.07.1973 or 22.7.1973
    @[word = "[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}/" ]
    |
    # two tokens: day.month ('\text{)}year
    # 22.07 '73 or 22.7. 1973
    [word = "[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{1,2}/" ]
    [word = "'" ]?
    @[word = "([0-9]{2}\.[0-9]{2}\.[0-9]{2}\.[0-9]{2}\.[0-9]{2}\.[0-9]{2}\.[0-9]{2}\.[0-9]{2}\.[0-9]{2}\.[0-9]{2}/" ]
    |
    # three tokens: day. month ('\text{)}year
    [word = "[0-9]{1,2}\.[0-9]{1,2}/" ]
    @[lemma = RE($\text{name\_month}$)]
  )
\end{verbatim}
The surface realization of dates is very variable. Thus, the kernel of date structures results in many disjunctions in the rule. Dates usually have three meaningful units: day, month and year. All of them can be represented by cardinal numbers. Day and month can consist either of one or two letters, years of two or four letters. The month can additionally be represented by a noun. The single units can be combined in different ways. Consisting solely of cardinal numbers, it can be represented as one token, separated only by dots, or as two tokens, day and month grouped together and the year separated (3.75).

(3.75) 2.11.1995

In the case that the month is represented by a noun, dates can consist of three tokens as in (3.76), or of two tokens as in (3.77), where the day unit is missing.


(3.77) Mai 1991 (May 1991)

Dates can be modified by a preceding weekday, optionally with or without the fixed determiner den (accusative the) as in (3.78).

(3.78) Sonntag, den 22. Juli 1973
       Sunday, the 22. July 1973

Coordination of day and month or both is also possible as in (3.79).

(3.79) a. 2. bis 3. Juli 1984
       2. thru 3. July 1984

3.3 Complex phrasal categories

NC rules for named entities comprise proper nouns, sports clubs and news agencies. The rule for proper nouns \texttt{nc\_ne} is given below.

\begin{verbatim}
MACRO nc_ne(0)
  (@[lemma = RE($noun\_Tite1)]? # title noun
   @[lemma = RE($noun\_Tite1)]? # proper nouns
   (
   # either all in capital letters
   [pos = "NE|FM|NN" & word = "[A-Z]*"]+ |
   )
   # or all upper case
   [pos = "NE|FM"
    | (pos = "NN" & alemma = "]")
    & word = "[A-Z] [a-z-]*([-A-Z] [a-z-]*)?"
    %d]
   |
   # or lower case first letter
   [pos = "NE|FM"
    & word = "[a-z-]*"
    %d]
  )+ |
  ( |
  # abbreviation of first name possible |
  [word = "[A-Z]\."] |
  [pos = "NE|FM"
   | (pos = "NN" & alemma = "]")
   & word = "[A-Z] [a-z-]*([-A-Z] [a-z-]*)?"
   %d]
  )*
  [pos = "NE|FM"
   | (pos = "NN" & alemma = "]")
   & word = "[A-Z] [a-z-]*([-A-Z] [a-z-]*)?"
   %d]
  )
  | # or all in capital letters
  [word = "[A-Z]*"]+
\end{verbatim}
Proper noun chunks consist of an unspecified number of adjacent proper nouns or foreign language material. As the tagging is problematic with respect to the correct identification of proper nouns, we impose constraints. Adjacent proper nouns have to fit together. In other words, the proper nouns have to be either all in capital letters (3.80), or all beginning with an upper case letter (3.81), or all beginning with a lower case letter (3.81).

(3.80) RIO DE JANEIRO

(3.81) Kathi Wilhelm

Words which are in written in capital letters are considered to be proper nouns, even if they are tagged otherwise. The same holds for unknown capitalized words. We assume that unknown capitalized words are more likely to be proper nouns. Proper noun chunks can be extended by a title noun (3.82).

(3.82) Professor Christian Rohrer

In this case, we assume that the proper noun modifies the title noun more closely.

The second rule for named entities is concerned with sports clubs nc_sport.

MACRO nc_sport(0)
(   [word = RE($noun_sportabbr) | word = RE($noun_sport)]+   [pos = "NE"]]*)
  @[pos = "NE"]
)@)[pos = "NE"]
);

The rule searches for an unspecified number of proper nouns which are preceded by specific nouns representing a sports club (3.83).

(3.83) VfB Stuttgart

The modifiers can be regular nouns or abbreviations.

The third rule nc_news for named entities is concerned with references to news agencies.
3.3 Complex phrasal categories

MACRO nc_news(0)
(
    # proper noun chunk
    /nc_ne[]
    ( 
        /braca[]
        # indefinite number of news agency abbreviations
        @[pos = "N[NE]" & lemma = RE( $noun_dpa )]
        [pos = "KON" | word = ",|\"]? 
    )
    /brace[]
)
;

The kernel of these noun chunks are adjacent nouns which refer to news agen-
cies ($noun_dpa). These nouns can be coordinated and are enclosed in paren-
theses. A proper noun chunk functions as modifier (3.84).

(3.84) CHRISTIAN DE VEGT ( dpa )

Three NC rules are concerned with NCs with numeral aspect. The first rule
nc_card is given below.

MACRO nc_card(0)
(
( 
    # rund 3 Millionen (round 3 million)
    [lemma = RE( $adv_grad )]? 
    [pos = "CARD"]? 
    @[lemma = RE($noun_Card)]
) 
| 
( 
    # beinahe 5 (almost 5)
    [lemma = RE( $adv_grad )]
    ( 
        @[word = "[0-9]1,3"]+ 
    )
    @[pos = "CARD"]
) 
) 
;

105
Grammar Details

Cardinal noun chunks consist either of a cardinal noun as head optionally modified in pre-head position by a cardinal number and/or a grade adverb (3.56, p. 96), or simply of a cardinal number modified by a grad adverb (3.85).

(3.85) beinahe 5 (almost 5)

The second rule nc_meas searches for measure noun chunks.

MACRO nc_meas() 
(
   /cardcoord_first[]?
   @[word = RE($noun_Measure)]
)
;

The kernel is a measure noun such as Glas (glass), which is optionally modified by cardinal numbers which can be coordinated (3.57+3.86).

(3.86) 3 bis 4 Glas (3 to 4 glas of)

Coordination of the cardinals is not limited to coordination with comma and coordinating conjunctions but includes the coordination with prepositions (4 bis 5; 4 to 5), possessive markers (4 meiner 5; 4 of my 5) and articles (4 der 5; 4 of the 5) as well.

The third rule nc_numb searches for nouns which can be modified by a cardinal in post-head position (3.58, p. 96).

MACRO nc_numb() 
(
   @[lemma = RE( $noun_precard )]
   [pos = "CARD"]
)
;

Two specialized rules are concerned with “locative” NCs. The first rule nc_tel is given below.

MACRO nc_tel() 
(
   @[word = RE($noun_telef)]
   ( [pos = "CARD"]+ 
    ( [pos = "KON" | word = "[,\/]" ] [pos = "CARD"]+ )*
   )
)
The kernel of this structure is a noun which refers to a telephone number, which is followed by a cardinal number, which can optionally be coordinated (3.60, 3.87).

(3.87) Rufnummer 0 61 92 / 1 11 03  
telephone number 0 61 92 / 1 11 03

The second rule nc_street, which refers to a street address.

MACRO nc_street(0)  
(  
    # street name  
    [pos = "TRUNC"]?  
    @(word = RE($noun_street)]  
    # house number  
    [pos = "CARD" & word = "[0-9-]*"]
  )

MACRO nc_trunc(0)  
(  
    # street name  
    [pos = "N[NE]"]  
    @(word = RE($noun_street2)]  
    # house number  
    [pos = "CARD" & word = "[0-9-]*"]
  )
The kernel is a noun which is obligatorily coordinated by an unspecified number of truncated elements (3.61, p. 96).

The two first level NP rules are listed in Table 3.21.

<table>
<thead>
<tr>
<th>rules</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>np</td>
<td>default NP</td>
<td>3.88</td>
</tr>
<tr>
<td>(np Coord)</td>
<td>coordinated NC or NP (used in np and pp macros)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.21: First level NP rules

The first rule np given below searches for simple NP structures beginning with a determiner, and ending with a NC structure.

MACRO np(0) ( /allDet[] /ac_coord[]? <nc> [ ]* @[._nc_f not contains "meas"] </nc> ) ;

The only modifiers are first level APs, which can optionally be coordinated (3.88).

(3.88) a. diese Fähigkeiten
        these capabilities

       b. die netten Studenten
        the nice students

The NC structure is restricted to NCs, which do not have a measure noun as head. This is due to the special property of measure nouns as potential pre-noun (Glas Bier glass beer), and pre-adjective modifier (Meter hohe meter high).
Often the preceding determiner does not belong to the measure NC but rather to a higher noun projection.

(3.89) a. \([NP \ \text{die} \ [NC \ \text{sechs Glas}] \ Bier]\)
    the six glass beer

b. \([NP \ \text{der} \ [AP \ [NP \ \text{drei Meter}] \ \text{hohe}] \ Baum]\)
    the three meters high tree

    the tree which is three meters high

Instead the correct attachment of the measure NCs to the AP or the NP are performed first.

The second first level NP rule \(\text{np}_\text{coord}\) is not annotated directly. It is used in NP and PP rules.

\begin{verbatim}
MACRO np_coord(0)
(
    (word = ",,"
    (/region[nc] /region[np])
)*
    [pos = "KON"]
    (/region[nc] /region[np])
)
;
\end{verbatim}

It searches for NC and NP conjuncts separated by commas or a coordinate conjunction.

The only pronominal NP rule is given in Table 3.22.

<table>
<thead>
<tr>
<th>rules</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>np_pron</td>
<td>pronominal NP</td>
<td>3.66, 3.67, 3.68, 3.69</td>
</tr>
</tbody>
</table>

Table 3.22: First level NPPRON rules

The rule searches for substitutive demonstrative pronouns, substitutive indefinite pronouns, irreflexive personal pronouns, substitutive possessive pronouns, substitutive relative pronouns, substitutive interrogative pronouns, and reflexive pronouns.

\begin{verbatim}
MACRO np_pron(0)
(

\end{verbatim}
The pronominal NPs are then differentiated by the Perl-scripts, and classified as interrogative (\textit{w}h)(3.68, p. 98), relative (rel)(3.69, p. 98), reflexive rel1(3.67, p. 98), and reflexive \textit{sich} rel1 \textit{sich}. The rest of the pronouns is not more closely defined. Additionally, all pronouns are marked by the feature \textit{pron}.

\textbf{Second level grammar rules}

There are two different kinds of second level noun chunk rules:

- rules for NPs without determiner (NCC rules)
- rules for NPs with determiner (NP rules)

The NCC rules precede the NP rules. They build up the main body of noun chunks. NP rules are responsible mainly for filling the position of the determiner. NP structures resulting from NCC rules are annotated as different structures during the annotation process to distinguish NPs with determiner (NP) from NPs without determiner (NCC). NCC structures can be seen as quasi bar-level projections, which are, however, annotated only temporarily. The introduction of a quasi bar-level projection for NPs has technical reasons. NCCs can still be extended in pre-head position either by additional structures, or by the same structure which has been enlarged. APs, e.g., can be included in an NCC structure at one stage as a simple structure. During the next run of the second level, the AP may have become a complex structure. The expanded AP automatically expands the NCC as well. NP structures, on the other hand can no longer be extended in pre-head position, but only at the end of the structure.

An overview of the second level grammar rules for NCCs is given in Table 3.23.

There are basically only two different rules for NCCs in the second level. The default rule \texttt{ncc}, and a rule for NCCs in quotation marks \texttt{ncc_quot}. Both rules have a variant which is valid during the first run of the second level. This variant is basically the same rule with a slight difference. NCC rules include NCC structures. NCC structures, however, are not built before the second level. Thus, during the first run of the second level, they are not available. The technical framework of YAC does not allow the inclusion of structures in the rules which are not yet available. Thus, NC structures replace the NCC structures in the first level NCC rule variants \texttt{ncc_first}, and \texttt{ncc_first_quot}.

The only obligatory element of the default rule \texttt{ncc} is its NC kernel.

\begin{verbatim}
MACRO ncc(0)
( 

110
\end{verbatim}
3.3 Complex phrasal categories

<table>
<thead>
<tr>
<th>ncc</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ncc_first</td>
<td>NP without determiner (first run of second level)</td>
<td></td>
</tr>
<tr>
<td>ncc</td>
<td>NP without determiner</td>
<td></td>
</tr>
<tr>
<td>ncc_first_quot</td>
<td>NP without determiner in quotation marks (first run of second level)</td>
<td></td>
</tr>
<tr>
<td>ncc_quot</td>
<td>NP without determiner in quotation marks</td>
<td></td>
</tr>
<tr>
<td>(ncc_gen_mod)</td>
<td>NP without determiner with the function of a genitive modifier</td>
<td></td>
</tr>
<tr>
<td>(ncc_incorr)</td>
<td>incorrect NPs without determiner (used in connection with NCC rules)</td>
<td></td>
</tr>
<tr>
<td>(ncc_secure)</td>
<td>secure context for AdvP attachment (used in NCC rules)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.23: Second level NCC rules

```plaintext
# optional adverbial modifier with secure context
(
    /ncc_secure[]
    /advp_mod[]+
)?
# optional cardinal modifier with optional coordination
/cardcoord[]?
# optional AP modifiers with optional coordination
/ap_coord[$$_.c]?$
# NC kernel
<nc
    @[...nc_f not contains "year"]
    []*
</nc>
# optional post-head nominal genitive modifier
/np_mod[]?
(/ncc_gen_mod[]|np_gen_mod[])*
/np_mod[]?
)
;

All NC structures except year dates can function as kernel of NCCs. This kernel can be modified in pre-head as well as in post-head position. Pre-head modifiers can be adverbial phrases, cardinal numbers, and APs. Adverbial modifiers are restricted to a reliable context ncc_secure, which is simply a preceding
preposition. In other words, when the NCC is embedded in a PP structure. Cardinal and AP modifiers can optionally be coordinated. In the case of APs, the coordinated APs have to agree with each other (cf. section 3.3.1).

Post-head modifiers can be NPs with genitive case, or named entities. Post-head genitive modifiers can be NCC and NP structures. The rule for post-head genitive NPs np_gen_mod is given below.

```plaintext
MACRO np_gen_mod(0)
(
    <np>
        [._np_agr contains "Gen:*"]
    </np>
)
;
It simply includes NPs containing genitive case. The rule for post-head genitive NCCs ncc_gen_mod is more restricted.

```plaintext
MACRO ncc_gen_mod(0)
(
    <ncc>
        [._ncc_f not contains "nogen"
         & _ap_f not contains "pp\np"
         & _ncc_agr contains "Gen:*"
         & pos != "CARD"]
    </ncc>
)
;
As NCCs consisting of a single noun are usually completely ambiguous with respect to case, they are excluded as genitive modifiers if their case is ambiguous. These single noun NCCs are marked with the feature nogen. Besides, the NCC must not begin with a cardinal number or an AP embedding PP or NP structures.

Named entities can be NCs in brackets or parentheses (brac), in quotation marks quot or (multi-word) proper nouns (ne), as well as NCCs and NPs in quotation marks (quot).

```plaintext
MACRO np_mod(0)
(
    <nc>[._nc_f contains "brac|quot|ne"][]</nc>
    |<ncc>[._ncc_f contains "quot"][]</ncc>

112
```
The named entities are identified by the feature attributes.

The secondary rule `ncc_incorr` is used in combination with the default rule ncc.

MACRO `ncc_incorr(0)`
```
<match>
  <nc>
    [*
      <target>
        [._nc_agr matches "Gen:*"
        & _nc_f contains "ne"]
      [*
    ]
  </nc>
```

It filters out NCCs beginning with a genitive NC kernel. These structures are not NCC structures but NP structures beginning with an NP possessive marker in the position of the determiner.

The second NCC rule `ncc_quot` is very similar to the default rule.

MACRO `ncc_quot(0)`
```
# optional adverbial modifier with secure context
(#optional adverbial modifier with secure context
  (ncc_secure[]
   /advp_mod[]+)
)?
# optional cardinal modifier with optional coordination
/cardcoord[]?
# possible begin of quotation
/quot[\$
# optional AP modifiers with optional coordination
/ap_coord[\$
# possible begin of quotation
/quot[\$
# optional AP modifiers with optional coordination
The main difference is that quotation marks enclose parts or all of the NCC structure. The quotation marks enclose the meaningful parts of the NCC. In other words, the beginning quotation mark can be at any point from preceding the first AP to preceding the NC kernel. The closing quotation mark is at the end of the structure. The enclosing quotation marks provide a reliable context to perform PP-attachment reliably. In this specific context PPs are attached to the preceding head noun.

An overview of the second level grammar rules for NPs is given in Table 3.24.

The default rule for second level NPs is np.

MACRO np(0)

( /alldet[]
 /region(ncc, ’@’)
 )

It includes different types of determiners and an NCC. The result are examples as in (3.90, Figure 3.11).

(3.90) das vorher von Stiefeln betretene Hemd
the before of boots stepped on shirt

the shirt which before was stepped on by boots

The post-processing distinguishes three different types of NPs depending on the determiner:

• wh-NPs (wh) beginning with an attributive interrogative wh-pronoun

114
3.3 Complex phrasal categories

<table>
<thead>
<tr>
<th>rules</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>np</td>
<td>default NP</td>
<td>3.90, Figures 3.11,</td>
</tr>
<tr>
<td>np_quot</td>
<td>NP in quotation marks</td>
<td>3.91, Figures 3.12,</td>
</tr>
<tr>
<td>np_poss</td>
<td>NP with prehead possessive NP</td>
<td>3.62</td>
</tr>
<tr>
<td>np_poss quotas</td>
<td>NP with possessive NP in quotation marks</td>
<td></td>
</tr>
<tr>
<td>(np_gen_mod)</td>
<td>NP as genitive modifier (used in ncc macros)</td>
<td></td>
</tr>
<tr>
<td>(np_coord)</td>
<td>coordinated NPs (used in pp and ap macros)</td>
<td></td>
</tr>
<tr>
<td>(np_coord_first)</td>
<td>coordinated NPs (first run of second level)</td>
<td></td>
</tr>
<tr>
<td>(np_mod)</td>
<td>NC, NCC and NP [brac,quot,ne] as post-head modifiers (used in NCC rules)</td>
<td></td>
</tr>
<tr>
<td>(np_mod_first)</td>
<td>NC, NCC and NP [brac,quot,ne] as post-head modifiers (used in NCC rules during first run of the second level)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.24: Second level NP rules

Figure 3.11: TIGERSearch tree diagram of (3.90)
Grammar Details

- relative-NPs (rel) beginning with an attributive relative pronoun
- default NPs (norm) beginning with any other determiner

The second NP rule np_quot is again very similar to the default rule.

```
MACRO np_quot() {
    /quota[$$a]? /alldet[] /quota[$$b]?
    <ncc> @/[xor[$$a, $$b]] [[]]*/ </ncc> /region(pp)? /quote[]
}
```

The main difference is the enclosing quotation marks. The beginning quotation marks can be either before or after the determiner. The closing quotation marks are at the end of the structure. Again, the quotation marks provide a reliable context which allows PP-attachment to perform reliably (3.91, Figure 3.12).

(3.91) \[
\begin{array}{c}
_{NP}\ \text{einem “durchgeknallten Ex-Army-Typ [}_{PP}\ \text{von der Air-Force]} \\
\text{a manic ex-army guy from the Air-Force}
\end{array}
\]

The third NP rule np_poss has a possessive NP at the position of the determiner.

```
MACRO np_poss() {
    ( <np>
        ( [...np_agr matches "Gen:*" & _.np_f not contains "pron"] |
            [ambiguity(_.np_agr) = 0 & word = ".*[s']" & _.np_f not contains "pron"]
        ) [[]]*/ </np>
}
```
3.3 Complex phrasal categories

The possessive NP is either a NP or NC structure which is not pronominal and is definitely genitive case (matches).

The forth NP rule np_poss_quot is similar to the rule nc_poss, but with enclosing quotation marks.
The enclosing quotation marks allow the relaxation of restrictions on the possessive NP or NC, and to allow PP-attachment.

**Third level grammar rules**

In the third level all of the three different annotated nominal structures (NC, NCC, NP) are collected, and re-annotated as NPs. Thereby, NCs remain annotated as NC as well. NC structures are kept for the final annotation as they incorporate important information about local structural peculiarities.

Aside of the simple collecting of structures, there are a few grammar rules executed during the third level. An overview of the third level grammar rules for NPs is given in Table 3.25

The first third level NP rule is `np_headless`.

MACRO np_headless(0)
(`
 /alldet[]
 <ap>
 [!np]
 []*
3.3 Complex phrasal categories

<table>
<thead>
<tr>
<th>np</th>
<th>rules</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>np_headless</td>
<td></td>
<td>headless NP</td>
<td>3.92</td>
</tr>
<tr>
<td>np_ne</td>
<td></td>
<td>possible named entities</td>
<td>3.93</td>
</tr>
<tr>
<td>np_ne2</td>
<td></td>
<td>possible named entities</td>
<td>3.93</td>
</tr>
<tr>
<td>(np_coord)</td>
<td></td>
<td>coordinated NPs (used in pp macros)</td>
<td></td>
</tr>
<tr>
<td>np_coord_annot</td>
<td></td>
<td>coordinated NPs</td>
<td>3.94</td>
</tr>
</tbody>
</table>

Table 3.25: Third level NP rules


Headless NPs are annotated after the main parsing process has been completed. Only then, can they be reliably identified. The rule simply searches for determiners followed by an AP which is not yet embedded in an NP (3.92).

(3.92) \[ NP \ \text{die} \ [AP \ jüngste ] \]  
the youngest

The fact that all NPs are already built, makes it possible to determine reliably that the determiner belongs to a headless NP with AP “kernel”.

Two of the third level NP rules are concerned with proper nouns. As is the case with quotation marks, and parentheses commas are used as reliable context to relax rules for proper noun structures. The assumption behind both rules is that NP structures enclosed in commas are possible proper noun structures even if the PoS-tag does not indicate this. We use this technique as a heuristic to overcome possible tagging errors. The first of these rules np_ne includes two or more adjacent NCC structures surrounded by commas.

MACRO np_ne(0) (  
    [lemma = ",," ]
    (  
        <ncc>  
            []
        </ncc>
    ){2,}  
    [ lemma = ",," ]
  )

The second rule np_ne2 is more complex.
In this case, the first NP structures, an NCC or an NP, have to end with a proper name base noun chunk, followed by an unlimited number of non-restricted NP structures, also either NCCs or NPs. The whole construction is surrounded by commas. Both rules are apt to find structures as in (3.93).

(3.93), Luis Donaldo Colosio,

The last two third level NP rules are concerned with coordination of NPs. The rule np_coord is not directly used to annotate a structures, but is used in other structures.

MACRO np_coord(0)
(
    (nu
        [word = "",""
        /region(np)
    )*,

    120
3.3 Complex phrasal categories

It includes the second part of a NP coordination, i.e., the coordinated elements. There can be an unlimited number of coordinated elements separated by a comma followed by a final coordinated element separated by a coordinating conjunction.

The second coordination rule for NPs np_coord_annot is used to annotate a coordination structure of NPs (CNP).

```xml
MACR0 np Coord annot (0)
(

<np>
  ![pp]
  [] *
</np>
(

  [word = "","]
  /region(np)
  )*
  [pos = "KON"]
  /region(np)
)
)
;
```

The rule used to annotate coordinated NP structures is basically the same as the rule np_coord, however, it additionally includes the first element of the coordination (3.94).

(3.94) der Schöne, der Trommler und das Biest
the beauty, the percussionist and the beast

### 3.3.3 Prepositional Phrases (PP)

Prepositional phrases (PP) are phrasal structures with prepositions or postposition as head. The head can be realized as a simple preposition, as a contraction of preposition and article (am; at the), or as a multi-word preposition (bis zur; up to). PPs can be simple pronominal structures, solely consisting of a pronominal adverbial (3.95a), embedding a simple NP (3.95b), or embedding an AdvP (3.95c). PPs, however, can also be complex phrasal structures embedding complex NPs (3.95d, Figure 3.13), or embedding coordinated NPs (3.95e, Figure 3.14)
(3.95) a. damit (with it); wofür (what for)

b. \[ PP \text{ am} \ [ NP \text{ Bahnhof } ] \]  
at the station

c. \[ PP \text{ von} \ [ AdvP \text{ dort } ] \]  
from there

d. \[ PP \text{ mit} \ [ NP \text{ kleinen } ] , \ [ AP \text{ über} \ [ NP \text{ die Köpfe} \ [ NP \text{ der} \text{ Apostel} ] ] \text{ gesetzten} ] \text{ Flammen] } \]  
with small above the heads of the apostles’ set flames

‘with flames set above the heads of the apostles’

e. \[ PP \text{ zwischen} \ [ NP \text{ Basel } ] \text{ und} \ [ NP \text{ St. Moritz } ] \]  
between Basel and St. Moritz

**Annotation scheme**

An overview of the annotation scheme of PPs is given in Table 3.26.

<table>
<thead>
<tr>
<th>feature value</th>
<th>CQP</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agr</td>
<td>pp_agr</td>
<td>agreement information</td>
</tr>
<tr>
<td>f</td>
<td>pp_f</td>
<td>lexical-semantic properties</td>
</tr>
<tr>
<td>h</td>
<td>pp_h</td>
<td>head lemma</td>
</tr>
<tr>
<td>hpos</td>
<td>pp_hpos</td>
<td>corpus position of head lemma</td>
</tr>
</tbody>
</table>

Table 3.26: Annotation scheme of PP (prepositional phrase)

The annotation scheme of PPs comprises structural information, head lemma, and lexical-semantic properties. The head lemma annotation includes the preposition and the head noun of the embedded NP or the head adverbial of the embedded AdvP respectively. The two lemmas are separated by a colon (am: Bahnhof). The head position of the PP is available during the parsing process in order to determine the hierarchical structure of the final annotation.

**Lexical-semantic properties**

Lexical-semantic properties annotated for embedded NPs or AdvPs are projected to the PP. An overview of the lexical-semantic properties annotated as feature attributes especially for PPs is given in Table 3.27.

PPs embedding AdvPs instead of NPs (3.96) are marked by the feature adv.
3.3 Complex phraseal categories

Figure 3.13: TIGERSearch tree diagram of (3.95c)
Figure 3.14: TIGERSearch tree diagram of (3.95c)

<table>
<thead>
<tr>
<th>values</th>
<th>description</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>adv</td>
<td>embedded adverb</td>
<td>3.96</td>
</tr>
<tr>
<td>fus</td>
<td>fusion of preposition and article</td>
<td>3.97</td>
</tr>
<tr>
<td>nonp</td>
<td>no embedded NP</td>
<td></td>
</tr>
<tr>
<td>norm</td>
<td>default</td>
<td></td>
</tr>
<tr>
<td>post</td>
<td>postposition</td>
<td>3.99</td>
</tr>
<tr>
<td>pron</td>
<td>pronominal</td>
<td>3.98</td>
</tr>
</tbody>
</table>

Table 3.27: Lexical-semantic properties of PP (prepositional phrase)
3.3 Complex phrasal categories

(3.96) \[ PP \text{ bis } \text{[AdvP heute]} \]
until today

PPs where the preposition is contracted with the article (3.97) of the embedded NP are marked by the feature \text{fus} (fusion).

(3.97) \[ PP \text{ zum } \text{[NP verantwortlichen Redakteur]} \]
to the responsible reporter

Pronominal PPs (3.98) are marked by the feature \text{pron}, and additionally by the feature \text{nonp}.

(3.98) dafür (for it)

The feature \text{nonp} is necessary to differentiate between PPs annotated with the feature \text{pron} because the embedded NP is pronominal, and real pronominal PPs.

PPs with postpositions (3.99) instead of prepositions are marked by the feature \text{post}.

(3.99) \[ PP \text{ [NP der Kinder] wegen }] \]
the children because of
‘because of the children’

The other PPs are marked by the default value \text{norm}.

First level grammar rules

An overview of the first level grammar rules of non-pronominal PPs is given in Table 3.28.

<table>
<thead>
<tr>
<th>rules</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>pp</td>
<td>default PP</td>
<td>3.100, 3.101, 3.103, 3.104</td>
</tr>
<tr>
<td>ppost</td>
<td>PP with postposition (possibly reinterpreted as default PP)</td>
<td>3.107</td>
</tr>
</tbody>
</table>

Table 3.28: First level PP rules

As can be observed there are only two first level rules for non-pronominal PPs. The first rule \text{pp} searches for PPs with a preposition as head.
The preposition can be realized as a "normal" preposition, as a preposition contracted with the determiner of the embedded NP, or a multi-word preposition. In the latter case, the lemma of the preposition is derived from the contracted form by substitution rules within the post-processing.

The PP can embed NC and NP structures (3.100), which can optionally be coordinated (3.101).

(3.100) \[ PP \ vor \ [ NP \ rund \ acht \ Jahren ] \]

\[ \text{ago} \quad \text{around eight years} \]

‘around eight years ago’
Allowing coordinations for NPs embedded in PP is subject to ambiguity. It is also possible that some of the potential conjuncts are not embedded in the PP as in (3.102).

(3.102) \[NP \textit{Männer} \ P_{PP} \textit{im Anzug}] \text{ und } [NP \textit{Frauen} \ P_{PP} \textit{im Abendkleid}]\]

However, these cases are relatively rare. The ambiguity in this case is not very strong. In other words, there are many more cases where the potential conjuncts really belong to the PP than there are cases, where they do not belong to the PP. Thus, we have decided to include coordination of NPs, as this heuristic produces many more correct analyses than false ones.

Besides, the PP can embed AdvPs as in (3.103).

(3.103) \textit{ab morgen} \text{ ‘from tomorrow on’}

PPs embedding AdvPs instead of NPs are marked by the feature $adv$.

PPs can also embed separable verbal particles (3.104).

(3.104) \textit{mit dazu} \text{ with to it}

The preposition can also be supplemented by a circumposition (3.105+3.106).

(3.105) \textit{aus verschiedenen Motiven und Blickwinkeln heraus} \text{ ‘out of different motives and viewpoints’}

(3.106) \textit{von überall her} \text{ from everywhere}

The second first level PP rule $pppost$ searches for PPs with a postposition tag (3.107).
However, not all postpositions tagged as such really are postpositions. Some are annotated with the wrong PoS-tag. The rule `pppost` searches for both real postpositions and prepositions which have falsely been assigned the PoS-tag of a postposition. The rule looks for NC and NP structures which can be optionally coordinated, and are followed by a postposition. If the postposition does not belong to the class of words which can only or almost only be used as postpositions, in other words, if the lemma of the postposition is neither `zufolge` (according to), `willen`, `halber`, `zuliebe` (for the sake of) the postposition can be followed by additional NC and NP structures, these NCs and NPs can also optionally be coordinated. If this is the case, we reinterpret the wrongly assigned postposition as prepositions.

Besides of non-pronominal PPs, pronominal PPs are annotated in the first level. The first level rules for pronominal PPs is given in Table 3.29.

<table>
<thead>
<tr>
<th>ppron</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>prppron</td>
<td>pronominal PP</td>
<td>3.108, 3.109</td>
</tr>
</tbody>
</table>

Table 3.29: First level PPGRON rules
The rule for pronominal PPs `pppron` simply searches for pronominal adverbs and adverbial interrogatives.

```
MACRO pp-pron(0)
  (  
    [pos = "PAV|P WAV"]
  )
;
```

(3.108) dafür (for it); deswegen (because of it)

(3.109) worüber (about what); wobei (with what)

**Second level grammar rules**

An overview of the second level grammar rules of PPs is given in Table 3.30.

<table>
<thead>
<tr>
<th>pp</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>pp</td>
<td>default PP</td>
<td></td>
</tr>
<tr>
<td>pppost</td>
<td>PP with postposition (may be reinterpreted as default PP)</td>
<td></td>
</tr>
<tr>
<td>(cpp)</td>
<td>coordinated PPs (used in ap macros)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.30: Second level PP rules

The name of the two first rules, `pp` and `pppost`, correctly suggests that the second level rules of PPs do not differ greatly from the first level rules. The only difference between the first level PP rule `pp` displayed above, and the second level PP rule `pp` displayed below is that instead of NC and NP structures, NCC and NP structures can be embedded.

```
MACRO pp(0) 
  (  
    /prep[]
    (  
      (/region(ncc)|/region(np))
      /np_coord[]?  
    )
    <advp>  
    []*  
    </advp> 
  )
```
The difference between the rules is more in the resulting structures, which can be more complex in the second level (3.110, Figure 3.15). As has been argued before, this is not due to the difference in the rules, but to the difference in the embedded structure.

(3.110) \([_{PP} \text{mit} \_{PP} \text{kleine} \text{über} \text{die} \text{Köpfe} \text{der} \text{Apostel} \text{gesetzten} \text{Flammen}]\]

'swith flames set above the heads of the apostles'

Figure 3.15: TIGERSearch tree diagram of (3.110)

The same holds for the second level rule \(\text{pppost}\), which also differs only in the embedded NP structures, i.e., NCC and NP instead of NC and NP.
3.3 Complex phrasal categories

The third rule cpp is not directly annotated in the corpus but used in other rules (e.g., in the AP rule ap_pp).

MACRO cpp(0)
(
  ## pp (with variable on preposition)
  <pp>
    $$:[pos = "APPR|APPRART|KOKOM"]
    []*
  </pp>
  ## optionally coordination
  (  
    ## any number of pps separated by commas
    ## sharing the preposition
    (  
      [word = ","]
      <pp>
        [pos = "APPR|APPRART|KOKOM" & word=$$.word]
        []*
      </pp>
    )*
  )?  
);
);

The rule includes PPs followed optionally by an unlimited number of PPs coordinated by a comma, and finally followed by an optional PP coordinated by a coordinating conjunction. All coordinated PPs must share the same preposition (3.111).

(3.111) [pp zur Schule ], [pp zur Arbeit ], [pp zu privaten oder kulturellen Terminen ]
to school    to work    to private or cultural meetings

Third level rules

An overview of the third level PP rules is given in Table 3.31.
Grammar Details

<table>
<thead>
<tr>
<th>PP</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>pp</td>
<td>default PP</td>
<td></td>
</tr>
<tr>
<td>pposf</td>
<td>PP with postposition (may be reinterpreted as preposition)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.31: Third level PP rules

The third level PP rules, pp and pposf, are similar to the first level and the second level PP rules, pp and pposf. Again the difference lies in the embedded NP structures. As the different NP structures in the third level are subsumed under the label NP, the only nominal phrase structure which has to be included is NP.

3.4 Subordinate clauses

Subordinate clauses are simple, mostly non-recursive, clausal structures, which are built by combining the annotated phrasal structures. In contrast to a full parse the structure of the clauses is not hierarchical but flat. We annotate both finite and infinitival subordinate clauses. The finite clauses have to be introduced by a subordinating conjunction (3.112a), while infinitival clauses may (3.112b) or may not (3.112c) begin with a subordinating conjunction.

(3.112) a. daß auf der internationalen Wasserstraße Donau auch Schiffe verkehren dürfen
           that on the international water way Donau also ships operate can
           that ships can operate on the international water way Donau as well

b. um zu gewinnen
   for to win
   ‘to win’

c. in einer eigenen Wohnung zu leben.
   in a own apartment to live.
   ‘to live in one’s own apartment’

Annotation scheme

An overview of the annotation scheme of clauses is given in Table 3.32.
Daß KOUS auf der internationalen Wasserstraße Donau auch Schiffe verkehren können.
The annotation scheme of clauses comprises structural information, head lemma information, lexical-semantic properties, and verbal lemma information. The head lemma, if it exists, consists of the subordinating conjunction. Additionally, the lemma information of the main verb of the clause is given as a separate attribute (vlem).

### Lexical-semantic properties

The lexical-semantic properties of clauses are rather functional attributes such as `finite` and `infinitival`. An overview of the lexical-semantic properties annotated as feature attributes for clauses is given in Table 3.33.

<table>
<thead>
<tr>
<th>cl feature values for f</th>
<th>values</th>
<th>description</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>comp</td>
<td></td>
<td>3.116</td>
</tr>
<tr>
<td></td>
<td>fin</td>
<td></td>
<td>3.113</td>
</tr>
<tr>
<td></td>
<td>inf</td>
<td>infinite</td>
<td>3.114</td>
</tr>
<tr>
<td></td>
<td>rel</td>
<td>relative</td>
<td>3.115</td>
</tr>
<tr>
<td></td>
<td>subord</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.33: Lexical-semantic properties of CL (clause)

All clauses are subordinate clauses, and as such marked by the feature `subord`. Finite and infinitival clauses are marked by the features `fin` (3.113) and `inf` (3.114) respectively.

(3.113) nachdem Katharina Bohn weggezogen war

after Katharina Bohn moved away had

after Katharina Bohn had moved away

(3.114) um zu gewinnen

for to win

to win

134
3.4 Subordinate clauses

Relative clauses (3.115) are marked by the feature rel.

(3.115) die es sich erlaubt hatten
who it themselves permitted had
‘who had permitted themselves’

Comparative clauses (3.116) are marked by the feature comp.

(3.116) als es Antworten gibt
than it answers has
than there are answers

Rules for subordinate clauses

An overview of the clause rules is given in Table 3.34.

<table>
<thead>
<tr>
<th>cl</th>
<th>description</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelClause</td>
<td>relative clause</td>
<td>3.115</td>
</tr>
<tr>
<td>SubordClause</td>
<td>default subordinate clause</td>
<td>3.113</td>
</tr>
<tr>
<td>SubordCompClause</td>
<td>comparative clause</td>
<td>3.116</td>
</tr>
<tr>
<td>InfinSubordClauseWithConj</td>
<td>ininitive subordinate clause</td>
<td>3.114</td>
</tr>
<tr>
<td>InfinSubordClause</td>
<td>ininitive subordinate clause</td>
<td>3.117</td>
</tr>
</tbody>
</table>

Table 3.34: Clause level CL rules

The basic rule for clauses in the YAC framework is the rule for relative clauses RelClause.

MACRO RelClause (o)
{
    /pct []
    [pos = "APPR"]?
    <np>
        [_.np_f contains "rel"]
        []*
    </np>
    (  
        /phrasal_adjct []
    |
        /region[np]
    )*
    <vc>
Grammar Details

It begins with an introducer, in this case a NP with the feature rel. This can be either a substitutive relative pronoun or a NP beginning with an attributive relative pronoun. After this introducer, there are optionally a unspecified number of adjuncts and NP structures. In the case of relative clauses, the adjuncts are restricted to phrasal structures.

MACRO phrasal_adjct(0)
{
 /region[pp]
 |
 /region[advp]
 |
 [pos = "ITJ"]
 |
 <np>
   [._np_f contains "temp"]
   []*
   </np>
}

Phrasal adjuncts can be PPs, AdvPs, interjections, as well as temporal NPs. After these optional structures, there has to be a verbal complex containing a content verb. The verbal complex can finally be followed by an optional reflexive pronoun sich (oneself), or again an unspecified number of phrasal adjuncts. In order to provide a reliable context, the whole clause has to be surrounded by punctuation marks (e.g., dots or commas). An example is given in (3.115, p. 135).

There are two rules for finite subordinate clauses. The first SubordClause is the default rule for finite subordinate clauses.

MACRO SubordClause(0)

136
The rule is rather similar to the rule for relative clauses (RelClauses). The introducer in this case, however, is a subordinating conjunction. The subordinating conjunction can be modified by certain elements such as, comparative particles (als daß; as that), adverbs (ohne daß; without that), and negative particles (nicht daß; not that). The adjuncts following the subordinating conjunction are no longer limited to phrasal adjuncts but can comprise clausal elements as well.

MACRO adjct(0)
( /phrasal_adjct[]
 | /nomappo[]
 | /soappo[]
 | <cl>
   [...cl_f contains "rel"]
   []*
 </cl>
)
These clausal elements can be nominal appositions separated by commas (e.g., *der Professor, Christian Rohrer*, the professor, Christian Rohrer), appositions introduced by the word *so* followed by a phrasal adjunct, and relative clauses. The end of the clause is again marked by an obligatory verbal complex containing a content verb, which can be optionally followed by the reflexive pronoun *sich* and an unspecified number of phrasal and clausal adjuncts followed by a punctuation mark. Finite clauses introduced by a subordinating conjunction do not require the reliable context of punctuation marks, however, the elements following the verbal complex are necessarily connected with a punctuation mark. An example is given in (3.113, p. 134).

The second rule for finite subordinate clauses *SubordCompClause* deals with comparative subordinate clauses.

The second rule for finite subordinate clauses *SubordCompClause* deals with comparative subordinate clauses.

The rule is similar to the first rule for finite subordinate clauses *SubordClause*. The difference is that the clause in this case is introduced by a comparative conjunction and has to be surrounded by punctuation marks. An example is given in (3.116, p. 135).

There are also two rules for infinitival clauses. The first rule *InfInfSubordClauseWithConj* deals with infinitival subordinate clauses introduced by a subordinating conjunction as in (3.114, p. 134).
Again, the rule is very similar to the finite clause rule SubordClause with the only difference of being introduced by a subordinating conjunction with zu (to) and infinitive.

The second rule for infinitival subordinate clauses InfinSubordClause finds infinitival clauses without subordinating conjunction.
The difference to the first rule for infinitival clauses (InfinSubordClauseWithConj) is that there is no subordinating conjunction introducing the clause. Consequently, the range of the clause has to be restricted by surrounding commas. Besides, the verbal complex is restricted to be explicitly infinitival. An example is given in (3.117).

(3.117) eine Einladung zu akzeptieren
an invitation to accept
‘to accept an invitation’

Verbal complexes in the first infinitival clause rule (InfinSubordClause) are not restricted to explicitly infinitival VCs. The reason for this is that we can only unambiguously identify those VCs as infinitival which include the particle zu (to). All other VCs are left underspecified with respect to finiteness or infiniteness. Subordinating conjunctions introducing infinitival clauses provide a secure context to determine the infiniteness of VC. Thus, we do not have to restrict the VCs to the secure cases with the particle zu. However, if the infinitival clause is not introduced by a subordinate clause, we have to restrict the VCs to the secure cases.
Chapter 4

Evaluation

4.1 Introduction

The evaluation of text analyzing tools is not a trivial question, especially, if the results should be comparable to the results of other tools. There are basically two major problems with respect to evaluation:

- what should be evaluated?
- what kind of gold standard should be used?

The PARSEVAL measures of crossing brackets, (labeled) precision, and labeled recall (Black 1992; Black et al. 1991) are traditionally used for the quantitative evaluation of parsers. The quality of these measures with respect to giving a good picture of the actual quality of the parser output, however, has been questioned (cf. Manning and Schütze 1999). Besides, the relationship between the PARSEVAL measures and semantic relations has been frequently criticized (Carroll et al. 1998; Magerman 1995; Bangalore 1997). The main problem is that text analyzing tools vary significantly with respect to the theory underlying the analysis, as well as with respect to the output they produce.

Thus, a variety of other evaluation measures have arisen, such as, dependency-based, valence-based, exact, or selective category match (sketched in (Carroll et al. 2002)). The relational evaluation scheme proposed by (Briscoe et al. 2002) evaluates parse selection accuracy on named grammatical relations between lemmatized lexical heads, i.e., not the structure itself but the correct assignment of grammatical relations to the lexical heads of structures is evaluated. (Kübner and Hinrichs 2002) describe the problems of PARSEVAL measures for partial parsers which prefer partial analyses over uncertain ones. Unattached phrases lead to high losses in precision and recall. Kübner and Hinrichs, thus, prefer a dependency based evaluation as opposed to a mere phrase structure evaluation.
The second problem is concerned with the gold standard itself. Usually, if there is an existing treebank, it is used for evaluation. However, the parsing tool and the treebank have to be theory-conformant, i.e., the theory underlying the treebank, and the theory underlying the parsing tool have to be the same, or at least similar. Although most treebanks aim at a theory-independent annotation scheme, this is almost impossible. A dependency parser, e.g., cannot be evaluated on a phrase-structure-based treebank, at least not without comprehensive conversion rules, and vice versa. Even if the underlying theories are similar, problems arise wherever there is a small difference.

The conversion between two formats poses problems. The TIGER graph representation used in the TIGER treebank (Brants et al. 2002; TIGER) and the syntactic analysis provided by the LFG grammar, e.g., are very similar at the level of functional/dependency structure, although there are many differences. Thus, a broad coverage LFG grammar was used as one of the tools for building the TIGER treebank. The output of the LFG grammar was manually disambiguated and then automatically transformed into the TIGER export format (cf. Zinsmeister et al. 2001; Zinsmeister et al. 2002). The automatic transformation seemed feasible because the relationships between the two formats seem systematic. However, Zinsmeister et al. (2001) report that subtleties had to be handled with care. Forst (Forst 2003) now wants to exploit the existing TIGER treebank as a test suite for the same LFG grammar. However, it was not possible to simply change the direction of the transformation rules. Instead he had to construct new conversion rules. The problem is that although the formats are similar on functional/dependency level, there are quite a number of differences as well. Cases, where one of the formats is less specific than the other can only be evaluated concerning those aspects present in both formats. Especially problematic are cases, where the analyses are not one to one but \( n \) to \( m \), i.e., where a category in one of the formats can belong to two different categories in the other format, which cannot be subsumed under a single category.

### 4.2 Evaluating a chunker

The question is whether to evaluate maximal chunks only or the whole hierarchical structure with all embedded chunks. For most chunkers this is not a question as they annotate flat structures only. Thus, they do not have embedded structures. But what about those chunkers which do annotate hierarchical structures? Should they annotate all structures or only maximal chunks? As we annotate hierarchical structures because we believe they provide necessary information, we also want to know how good they are. Thus, we have to evaluate all structures, whether they are embedded or not.

However, the problem then is, that one loses the comparability to chunkers evaluated on maximal chunks only. Evaluations based on maximal chunks can-
4.2 Evaluating a chunker

not be compared to evaluations based on the whole constituent structure. The basis of the evaluation is simply not the same. False analyses of parts of a chunk can affect the whole hierarchy, especially if the maximal chunk is false. In this case, embedded structures are usually also false. Besides, it is not necessarily the case that a correct maximal chunk entails correct embedded structures. Usually, it is more difficult to deliver the correct internal structure than to deliver the correct maximal chunk. Thus, we make both evaluations:

- an evaluation of maximal chunks only, and
- an evaluation of all chunks, i.e., the full hierarchy of chunks

However, even then, it is difficult to compare the evaluation results of two chunkers. Chunkers deviate to a great extent with respect to the structures they annotate. Classical chunkers annotate base chunks only, i.e., flat non-recursive kernels of phrases. Other chunkers extend the chunk definition. They include some embedded structures in pre-head position, which can in some cases include recursive embedding. Some chunkers also include (recursive) embedding in post-head position as long as it does not produce ambiguities, i.e., PP-attachment is usually excluded. The difference is usually caused by the needs of the further application the chunker is built for. However, only chunkers with similar output, i.e., similar underlying chunk definition, and thus, similar structural ranges, can really be compared. Classical chunkers, e.g. cannot be compared with partial parsers or recursive chunkers as the task of the latter is considerably more difficult.

A problem for chunking approaches is also how to evaluate. For chunking approaches there is no real alternative to a phrase structure based evaluation. Neither relational evaluation nor dependency based evaluation are feasible as grammatical relations and dependencies are not annotated as such.

There are basically three different approaches:

- manually check the output of the chunker, and count the numbers necessary for the evaluation
- manually construct a gold standard corpus, which includes the necessary structures
- take an existing treebank as gold standard

The first and the second approach both have the advantage that they can be adapted to the underlying chunk definition. The first approach, however, is the least desirable as the evaluation in this case cannot be automatized for a new version of the chunker. The second approach has the advantage that once the gold standard is constructed, the evaluation can be repeated as often as desired. However, the construction of such a manually corrected corpus is
Evaluation
time consuming and costly. Thus, if this approach is taken, usually only a small
data set is made available to keep the manual work to a minimum. Another
disadvantage is that whenever the chunk definition is changed, the reference
corpus has to be checked and modified manually again.

Thus, it is most desirable to use an existing treebank. It has the advantage,
that the manual work has already been performed. Besides, treebanks are built
as general reference corpora for various applications. Thus, they usually are of
a considerable size. The problem in this case is that the annotation of a phrase
structure based treebank is a full hierarchical representation. The structures in
the treebank are deeper, and more hierarchical than the structures annotated
by chunkers. Thus, the structures of the treebank have to be broken down to fit
the underlying chunk definition of the respective chunker. The advantage
is, that if the structures are broken down automatically, the procedure can be
repeated with modified rules, when the underlying chunk definition is changed.

It is usually easy to extract the base chunks for the reference of a classical
chunker. Base chunks are often part of the annotation of treebanks. The refer-
ence of statistical parsers trained on part of the same treebank, or a treebank
with the same annotation scheme is also easy to extract as the chunks anno-
tated by these chunkers can be found as structure nodes in the treebank. The
same holds for symbolic chunkers which share the same annotation scheme.
However, if the annotation scheme of the chunker differs from the annotation
scheme of the treebank, it is difficult to extract the desired gold standard. In
this case, the structures annotated by the chunker do not correspond directly to
nodes in the treebank. Moreover, the structures are parts of nodes, and combi-
nations of embedded nodes (terminal and non-terminal). In some cases, parts
of the combined nodes have to be cut off.

Problems are also caused by differences in theoretical assumptions, e.g.,
although the NEGRA treebank (NEGRA Project) claims to be theory neutral, it
includes some theoretical peculiarities. The examples in (4.1) show the correct
hierarchical annotation of two NPs as it is produced by YAC.

(4.1) a. \([_{NP} \text{Ethno- und Dokumentarfilmer} \quad _{NP} \text{Echevarrea}]\)

\[\text{Ethno- and documentary film maker} \quad \text{Echevarrea}\]

b. \([_{NP} \text{David Harrington} \quad _{NP} \text{(violin)}]\)

The analysis of the NP in (4.1a) underlies the assumption that the em-
bedded NP \textit{Echevarrea} modifies the preceding nouns more closely. Within the
NEGRA analysis, this decision is not made. Instead, the elements \textit{Ethno- und}
\textit{Dokumentarfilmer} and \textit{Echevarrea} are both embedded in a flat structure under
the NP as elements of the noun kernel (edge label \(\texttt{NK}\)) (Figure 4.1). The two
desired gold standard NPs according to the theoretical assumption underlying
YAC as in (4.1a) cannot be extracted as such. The maximal NP is extracted cor-
correctly. However, not the embedded NP \textit{Echevarrea}. Instead, the CNP in Figure
4.1 is incorrectly extracted as embedded NP.
4.3 How to get a real gold standard

4.3.1 A German treebank: NEGRA

We decided to extract a gold standard from an existing treebank for German, the NEGRA treebank (NEGRA Project). The annotation format is described in (Skut et al. 1997). The structures are annotation graphs, which crossing edges to capture long distance relations. Grammatical relations are expressed by edge labels. We chose the NEGRA corpus instead of the TIGER corpus as the latter is still work in progress and was not finished when we started the evaluation.
Evaluation

We use the TIGER-XML export format (Mengel and Lezius 2000; Lezius 2002) as a basis to extract the reference, as it contains all available relevant information about tokens, and the hierarchical structure, and can easily be processed using XSLT-style sheets.

Collecting non explicit NPs

The annotation of the NEGRA corpus has some peculiarities that one has to deal with when building a gold standard.

- unary nodes are not explicit
- NP nodes under PPs are not explicit
- elements, which actually are not NPs, are subsumed under NP nodes

Thus, in order to extract all NPs, some nodes have to be inserted, and others have to be removed.

All single-item NPs are not projected to NP nodes, i.e., single-item nouns proper nouns, as well as pronominals of all kinds. They are annotated with PoS-tags only and marked with the respective grammatical relation they hold. Thus, it is necessary to list all the possible PoS-tags and/or grammatical relations and collect them as NPs. Some of them can be identified by the PoS-tag, some by the grammatical relation, and others by combinations of the two.

More complicated are NPs embedded in PPs. Elements of the NP kernel under an NP or PP node, are marked in the NEGRA corpus by the edge label \( \text{NK} \) (for noun kernel). This includes determiners, APs, and the head noun. All of these elements have to be collected and subsumed under an NP node. However, this subsumes also some elements which cannot be considered as NPs:

- pronominal adverbial replacing a PP
- adverbs or adjectives embedded in a PP
- relative clauses embedded in a PP

These false NPs have to be sorted out.

The annotation of the AdvP in Figure 4.3 (translation: 4.2) is also not straightforward.

(4.2) eher eine Minute als eine Sekunde
     more likely a minute than a second

It is, however, impossible to exclude these structures from the reference.

Problematic also are cases as described in (4.1a, Figure 4.1), where two different nominal NPs are annotated in a flat hierarchy. As described above,
4.3 How to get a real gold standard

Figure 4.3: NEGRA analysis of (4.2)
we assume that in this case the nominal element *Echevarrea* is an embedded NP modifier of the preceding nominal elements *Ethno- und Dokumentarfilmer*. The two different nominal elements, however, cannot be differentiated. Consequently, the NP *Echevarrea* failed to be extracted.

Another problem are structures which are annotated as NPs within the NEGRA treebank, but which would not be classified as NPs by most other theories. Examples are the embedded NP structure in (4.3a) and the NP in (4.3b). Both are annotated as NPs within the NEGRA framework. However, they are in fact comparative phrases introduced by the preposition *wie*.

(4.3) a. \[ NP \text{ dem gleichen intellektuellen und emotionalen Stand } [NP \text{ wie Menschenfresser und Kopfgeldjäger }] \]
\[ \text{man-eaters and bounty hunters} \]

b. \[ NP \text{ wie “Extras” und “Graffiti” } \]
\[ \text{as “extras” and “graffiti”} \]

In the YAC framework they are annotated as PPs, and only the NPs embedded in this phrase as NPs. Thus, we have to cut off the preposition at the beginning of these structures.

Multi-word proper nouns have a separate node label in NEGRA, *MPN*. These structures are usually NP structures, and we, consequently collect them as such. However, in some cases, the MPN are only part of the actual NP as in (4.4a), in other examples (4.4b) the multi-word proper noun (here: *Helter Skelter*) constitutes the whole NP embedded in a larger NP.

(4.4) a. \[ NP \text{ die New York Times } \]
\[ \text{the New York Times} \]

b. \[ NP \text{ dem Lied } [NP \text{ Helter Skelter }] \]
\[ \text{the song Helter Skelter} \]

As Figure 4.4 shows, in the NEGRA representation the MPN node encloses only the actual multi-word proper noun of the example in (4.4a). The problem is, that this MPN cannot be distinguished from the multi-word proper noun in Figure 4.5, where it constitutes a whole NP.

**Cutting out the reference data**

After we collected all the NPs from the reference corpus, we tried to cut out the syntactic fragments covered by YAC. We cut off all attachments not included in the underlying chunk definition of YAC, including the following:

- PP-attachment
4.3 How to get a real gold standard

Figure 4.4: NEGRA analysis of (4.4)

Figure 4.5: NEGRA analysis of (4.4b)
Evaluation

- post-head adverbial modifiers
- pre-head adverbial modifiers
- relative clauses
- clausal modifiers
- appositions

All these attachments are based on highly ambiguous decisions, which are not performed within the framework of YAC (as described in section 2.2).

The problem is that because of the crossing edges, the elements do not necessarily occur in linear order. Thus, it is not trivial to cut something off at the end of a structure. In most cases, however, the elements listed above can be uniquely identified by their node or edge label. Thus, we simply cut off all elements with a specific edge or node label, knowing that they occur at the beginning or the end of the structure only.

The consequence, however, is that we cannot take those attachments into account, which we annotate in secure contexts only. We, e.g., perform PP-attachment if the part of the NP including the PP is surrounded by quotation marks as in (4.5).

(4.5) \[[_{NP} \text{"Abenteuerreise [}_{PP} \text{ins Ungewisse "]}]]

“adventure travel into the uncertain”

However, as we cannot distinguish these PPs, from other modifying PPs in the NEGRA annotation we have to cut these PPs off in our annotation as well. The same holds for adverbial modifiers of NPs, which are also annotated in secure contexts but cannot be distinguished from other adverbial modifiers. Thus, all adverbial modifiers at the beginning of NPs are cut off in the corpus annotated by YAC.

XSLT-stylesheets, although useful for most operations on XML, are not good at handling recursion. Thus, we can only control elements embedded directly, but not those which are embedded deeper in the structure. PPs embedded in post-head NP genitive modifiers (4.6, Figure 4.6) pose a problem.

(4.6) \[[_{NP} \text{das Urteil} \quad [_{NP} \text{eines Amtsgerichtes [}_{PP} \text{in Flensburg ]}]]

the judgement of a district council in October

\[[_{PP} \text{im Oktober des abgelaufenen Jahres }]]

of the past year

The PP \textit{im Oktober des abgelaufenen Jahres} can be cut-off without recursion as it is directly embedded in the maximal NP. However, the PP \textit{in Flensburg} is not directly embedded in the maximal NP, but in the NP which functions as genitive modifier of the maximal NP. In order to eliminate both PP modifiers, the rule
4.3 How to get a real gold standard

Figure 4.6: YAC analysis of (4.6)
for cutting-off these structures has to be performed recursively. The same holds for cases with deep embedding of other elements such as relative clauses, or clausal appositions.

Even more problematic are adverbial modifiers at the beginning of NPs which are embedded in APs (4.7).

\[(4.7) \ [NP [AP furchtbar militaristische ] Trickaufnahmen ] \]

awfully militaristic animations

It is not feasible to cut off these structures, as they cannot be uniquely differentiated from other modifiers within the AP.

Another problem concerns coordinated structures. YAC performs coordination only on the level of maximal constituents or implicitly if the structures are embedded in another structure. Thus, coordinated NP have to be split into their conjuncts. Problematic are structures where there is a modifier which refers to all conjuncts (4.8a).

\[(4.8) \ a. [NP der 27-jährige [CNP Maler, Musiker und Komponist ]]
the 27-year-old painter, musician, and composer

b. [NP der 27-jährige Maler ], [NP Musiker ]
the 27-year-old painter, musician, and composer
und [NP Komponist ]

As YAC does not perform coordination on this level, and as the correct attachment of the modifiers is subject to ambiguity, we had to weaken the structural representation of NEGRA, i.e., we had to underspecify dependencies. Thus, we have to find a way to convert the analysis in (4.8a) to the analysis in (4.8b). In the case of CNPs embedded in NP or PP structures, we attach all noun kernel elements preceding the CNP to the first conjunct, and all noun kernel elements following the CNP to the last conjunct.

Another problem are coordinations involving truncated elements. We decided to annotate these structures as a single NP structure, rather than as a coordinate structure, as the first conjunct delivers only part of its meaning.

\[(4.9) \ [NP Ethno- und Dokumentarfilmer ]
Ethno- documentary film maker

As these structures are annotated as CNP in the NEGRA corpus, the elements are split into their conjuncts. Thus, in this case, we have to prevent these structures from being split, and collect them as NP structures instead.

A problem which we cannot solve are multiple-post-head genitive modifiers. In NEGRA, these modifiers are hierarchically ordered as in (4.10, Figure 4.7).
Within the framework of YAC, however, the genitive modifiers are not hierarchically ordered but annotated flat, in linear order. The respective graph of the example in (4.10) in the YAC representation is given in Figure 4.8 (p. 155). It is obvious that the two analyses are incompatible. Thus, although the YAC analysis is correct from a flat annotation point of view, we get false positive NPs (here: der bestehenden Wärmeschutzverordnung), and false negative NPs (here: der bestehenden Wärmeschutzverordnung des Jahres 1977), because YAC annotated the post-head genitive modifiers as two separate NPs, whereas in the NEGRA representation the second genitive NP modifies the first genitive NP. We could have written complex rules to convert the analysis in Figure 4.8 to the structural representation in 4.10, but this would have been a re-analysis. A re-analysis, however, is likely to be biased. Thus, we had to cut off post-head genitive modifiers in both analyses.

The resulting reference we were able to extract cannot really be called a gold standard. We were not able to extract all relevant NPs correctly. Consequently, the extracted reference contains some undesired NPs, and misses some desired ones. In some cases, we had to cut-off correct structures from the YAC annotation because we could not reliably extract the corresponding NP out of the NEGRA corpus. The reference does not reflect the full range of structural possibilities that YAC offers. Thus, the precision and recall figures we calculated can only give a sketchy image of the real performance of YAC.

This is even more so, as it is well known that even the annotation of a manually corrected treebank such as NEGRA is subject to faults and inconsistencies caused by human annotators. Brants (Brants 2000) reports an F-score between two annotators of 92.43% and an F-score of ca. 95% between the initial and the final version of the annotated corpus. Thus, even a manually corrected corpus cannot be expected to be 100% correct. A simple example for an incorrect tag on token level in the NEGRA treebank is the annotation of the noun Reifen (tire) as finite Verb (WFIN) in the example in 4.11.

(4.11) Gelegentlich muß sie den VW-Bus mit dem platten Reifen verlassen

‘Sometimes she must leave the VW-van with the flat tire’

As it was already very difficult to extract a more or less good reference for NPs from NEGRA, we did not try to extract references for other phrasal constructions. Instead we decided to opt for another approach to a gold standard.
Figure 4.7: NEGRA analysis of (4.10)
Figure 4.8: YAC analysis (4.10)
Evaluation on the NEGRA reference

For the evaluation of YAC against the reference extracted from the NEGRA corpus we use the manually disambiguated part-of-speech tagging provided. Additionally, we added lemma and agreement information from the IMSLex morphology (Lezius et al. 2000; Lüdeling and Fitschen 2002). This information was not disambiguated or corrected manually. In other words, the availability of ideal PoS-tags does not necessarily mean that the agreement is ideal as well. Unknown words do not get agreement information at all. If an unknown word has a homograph which is in the lexicon, the unknown word gets the agreement information of the known word. This can lead to wrong agreement information. The word *Schemen* can stand for both the plural of *Schema* (scheme) and the plural of members of a certain people *Scheme*. The former is part of the lexicon, the latter not. Thus, we had to deal with ambiguous morpho-syntactic information.

The 99,116 NPs in the reference set were then compared to the 101,165 NPs identified by YAC. Of those, 89,237 were correct (true positives), corresponding to a precision of 88.21% and a recall of 90.03%. This evaluation strategy is equivalent to computing labelled precision and labelled recall restricted to NP chunks. Since the head lemmas of NPs are not explicitly annotated in the NEGRA treebank and would have to be guessed from the part-of-speech tags at token level, we did not evaluate the head lemma annotations provided by YAC.

In real-life applications manually corrected part-of-speech tagging is not available. For this reason, we also evaluated YAC on a version of the corpus that was automatically part-of-speech tagged with the TreeTagger (Schmid 1994; Schmid 1995; TreeTagger ). We use the standard training corpus of the TreeTagger. The lexicon is customized based on the IMSLex morphology, i.e., all words in the corpus are extracted and looked up in the IMSLex. Thus, the tagging quality depends on the quality of the lexicon. With this set-up the TreeTagger achieved a tagging precision of 94.82% (336,692 tokens correct out of 355,096). Most of the tagging errors were proper nouns that were not correctly identified by the morphology.

With this fully automatic process, YAC identified 101,484 NPs, of which 85,353 were correct. This gives a precision of 82.48% and a recall of 86.11%. The decrease in precision and recall corresponds approximately to the error rate of the tagger.

YAC was not developed and tested on the NEGRA treebank. Therefore, evaluating YAC on this corpus shows its realistic performance on German newspaper text. The rules of YAC allow it to be easily adapted to specific text types. Thus, it would have been possible to optimise the rules for the specific characteristics of the NEGRA corpus. We deliberately did not do so, as we want to present a tool that is useful for extraction processes on unrestricted text.

Therefore, we believe the overall system performance (combining TreeTag-
4.3 How to get a real gold standard

er, IMSLex, and YAC) with a precision and recall close to 80% to be a good indicator of the usefulness of our tool. Especially, as we think that the problems connected with the extraction of the reference have a negative impact on the results, an assumption we tested making an evaluation on a manually built reference corpus.

4.3.2 A manually annotated corpus

In order to test real performance, we decided to build a second reference: a manually annotated corpus. As discussed before, the size of this corpus is by far smaller than the size of the first corpus. We extracted 400 sentences from the NEGRA corpus at random. The resulting reference contains 1920 NPs. We think that the size of the reference is large enough to make valid statements about precision and recall of the system. We calculated the precision as the number of true positives divided by the number of structures found by YAC, and the recall by dividing the number of true positives by the number of structures in the reference.

The big advantage of the manually corrected reference is that the annotated NPs can take the full range of structures into account which follow the underlying chunk definition of YAC. Thus, we can include all those structures which we had to exclude before, because of theoretical differences with the NEGRA annotation, which we could only solve by cutting off certain structures.

As the chunk definition of YAC is rather simple, there were no difficulties in deciding on the range of correct structures. The manually built reference should, thus, provide a good gold standard for the evaluation.

We built references for NPs, PPs, APs, and VCs. We did not evaluate AdvPs as the structure is so simple that it does not provide much information about the performance of the chunker. False analyses in this case are most likely to be caused by false analyses from the tagger providing the wrong PoS-tag leading to an incorrect identification. The results for each phrasal category were evaluated separately. The evaluation was performed on ideal PoS-tags taken from the NEGRA corpus as well as on automatically annotated PoS-tags produced by the TreeTagger. In both cases, the morpho-syntactic information was not ideal, but automatically added and left ambiguous.

Evaluation on ideal PoS-tags

Table 4.1 gives the evaluation figures of YAC on ideal PoS-tags.

Precision figures for the evaluation of all chunks range from approximately 96% for APs and NPs to 98% for PPs. As VCs are not recursive, they do not occur in this category. Recall is slightly lower ranging from 96.5% for APs and NPs to 97.5% for PPs. The figures are slightly lower for the evaluation of maximal chunks. The reason for the decrease of figures can be explained
Evaluation

<table>
<thead>
<tr>
<th></th>
<th>all chunks</th>
<th>maximal chunks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>precision</td>
<td>recall</td>
</tr>
<tr>
<td>NP</td>
<td>96.36</td>
<td>96.51</td>
</tr>
<tr>
<td>PP</td>
<td>98.08</td>
<td>96.51</td>
</tr>
<tr>
<td>AP</td>
<td>96.39</td>
<td>97.50</td>
</tr>
<tr>
<td>VC</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 4.1: Evaluation figures of YAC on ideal PoS-tags

by the fact that some structures are not combined to one large structure but left separate instead. In the case of NPs, e.g., certain post-head modifiers are not identified as such. Consequently, the maximal chunk cannot be identified correctly. However, embedded chunks can still be correctly identified. In the case of nominal post-head modifiers, the two NP structures are annotated as two separate maximal chunks as in (4.12a) instead of as one maximal chunk as in (4.12b).

(4.12) a. \([NP \text{ eine Art} \] [NP Gesundheitspolizei ]
   a kind of health police

   b. \([NP \text{ eine Art Gesundheitspolizei} ]

As in the example above, some of the post-head modifiers which are not identified as such are lexically selected by the preceding noun. Thus, without the necessary lexical information, the chunk cannot be correctly built. We have implemented some general selectional preferences in the grammar of YAC, e.g., for post-head modifiers of temporal nouns (4.13).

(4.13) \([NP \text{ Jahre } [AdvP später ]]
   years later

However, for other selectional preferences we do not yet have the necessary lexical information. Besides, if they are too specific with respect to their distribution or occur very seldom only, it might not be reasonable, for efficiency reasons, to implement them. We would need a large amount of specific rules which would cover only very few cases.

Precision and recall figures are quite similar for AP and NP which suggests that the annotation task is similarly difficult. Most of the false analyses of NPs are due to missing or false attachments of NP. For example, NPs with the function of possessive markers as in (4.14) failed to be identified as such.

(4.14) \([NP \text{ [NP Ludwig van Beethovens] Fantasie ]]
Ludwig van Beethovens GEN Fantasia

‘the Fantasia of Ludwig van Beethoven”
Remember that we operate on ideal PoS-tags but not on ideal morpho-syntactic information. Thus, often the genitive case is not identified correctly.

For VC structures, the figures are considerably higher than for the other structures. There are only very few mistakes in the annotation, which supports the assumption that VC structures are simple structures. The figures prove also that the simple approach to VCs combining any adjacent verbal elements independently of their character is sufficient to provide useful chunks. Thus, a more complex modelling does not seem necessary.

Evaluation on automatically annotated PoS-tags

In order to test the performance of YAC under real-life conditions, we evaluated its performance on automatically annotated PoS-tags as well. The evaluation figures of YAC on automatic PoS-tags are given in Table 4.2.

<table>
<thead>
<tr>
<th></th>
<th>all chunks</th>
<th>maximal chunks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>precision</td>
<td>recall</td>
</tr>
<tr>
<td>NP</td>
<td>89.93</td>
<td>91.67</td>
</tr>
<tr>
<td>PP</td>
<td>94.05</td>
<td>89.67</td>
</tr>
<tr>
<td>AP</td>
<td>84.24</td>
<td>89.25</td>
</tr>
<tr>
<td>VC</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 4.2: Evaluation figures of YAC on automatic PoS-tags

As can be observed, precision and recall figures for NPs and APs drop significantly if YAC is applied to a corpus PoS-tagged by the TreeTagger. The recall, although it is also significantly lower, is not affected to the same extent. Looking at the false analyses, it can be observed that two factors are responsible for most of the errors:

- proper nouns
- capitalized words

The first major factor for false analyses were proper nouns. In contrast to many other languages, German capitalizes both common nouns and proper nouns. Thus, it is difficult to distinguish between the two categories. The tagger has to guess whether an unknown capitalized word is a noun or a proper noun. In many cases, proper nouns are mistakenly tagged as common nouns. Thus, special rules for proper nouns cannot fire in all necessary cases. The consequence is that multi-word proper nouns cannot be assembled, and are left as separate NPs (4.15a). This is often punished not only once but several times, especially, if the proper nouns are embedded in larger structures. The correct analysis with flat annotation of post-head modifiers is given in (4.15b).
In this case, the tagger failed to identify the surname Völker as a proper noun. Consequently, both the maximal NP and the NP Professor Klaus Völker could not be identified correctly. This results in two false negatives and three false positives. Thus, one false PoS-tag entails several faults in the chunker output.

The second major factor for errors in the annotation are capitalized words. The tagger did not seem to take sentence boundaries into account. Thus, in many cases, capitalized words at the beginning of sentences were erroneously tagged as nouns. Most of these wrongly tagged capitalized words were adjectives, which explains the low precision figures of APs on automatically tagged text. Again, similar to the proper nouns, one false tag entails several false structures. As the adjective is mistakenly analysed as separate NP as in (4.16).

The result of this multiple punishment is that the precision for APs and NPs drops quite dramatically. The recall of APs is also largely affected as many adjectives are erroneously tagged as nouns. The recall for NPs drops less dramatically as the tagging errors affect fewer structures than the mere precision figures would suggest.

PPs are affected less by tagging errors. The recall figures, which are lower in comparison to the recall figures for ideally tagged text, suggest that more correct PPs failed to be identified. The reason is that words which can function as prepositions, can have other functions, and thus, other PoS-tags as well (e.g., conjunction). However, if a preposition is erroneously annotated with another PoS-tag, the PP can no longer be identified. Additionally, the precision of PPs can be affected by wrong NP assignment as well.

VCs are affected least by automatic tagging. The tagger can obviously identify verbal elements with a good precision. If the tagger makes errors, it is with respect to the character of the verbal element (e.g., finite vs. infinite). However, as the rules for VC leave the verbal elements underspecified, the chunking performance is not affected by assignment of the wrong kind of verbal element.
4.3.3 Discussion

As discussed above, the extraction of an NP reference from the NEGRA corpus was not really successful. Structures annotated by YAC had to be stripped of certain structures, because we could not filter the correct reference for the complete structure. The structures extracted from the NEGRA corpus in quite a number of cases are not correct. In other words, the NEGRA reference does not represent the theoretical assumptions underlying the annotation of YAC.

After examining the NEGRA reference, we had the impression that the performance of YAC must be better than the precision and recall figures suggested. Thus, we carefully annotated a manual reference to see whether our assumption was right. The evaluation figures we obtained from the manual reference were significantly higher than the ones obtained from the NEGRA reference. The evaluation figures for automatically tagged text obtained from the manual NP reference (precision: 89.93%, recall: 91.67%) are even higher than the figures for ideal tags obtained from the NEGRA reference (precision: 88.78%, recall: 82.36%). The respective figures for ideal tags obtained from the manual NP reference are considerably higher (precision: 96.36%, recall: 96.51%). This confirms our assumption that the evaluation on the NEGRA reference gives a poor picture of the actual performance of YAC. As the evaluation figures obtained from the manual reference give a much better picture of the performance of YAC, we will refer to these figures in the following.

4.4 Comparing different chunkers and partial parsers

As said before, comparing different chunkers and/or partial parsers is not straightforward. The underlying theoretical assumptions vary too much. It is neither useful nor does it make sense to compare base chunkers with more elaborate chunkers such as YAC. In order to compare tools, the underlying theory has to be comparable as well.

Another problem is, that most of the more elaborate chunkers and partial parsers for German do not provide evaluation figures. The reason is obviously the lack of a gold standard. As discussed above, it is even more difficult to get a gold standard for chunkers and partial parsers. It is very difficult and insecure to extract a reference from an existing treebank which has a full hierarchical representation. As we know, there is no treebank with partial analyses. Consequently, it is almost inevitable to construct a reference manually, which is always time consuming and expensive.

The only German chunker we know of, which goes beyond base chunks, and which provides evaluation figures is the chunker of Michael Schiehlen (Schiehlen 2002). He discusses the evaluation figures for both a base chunking
Evaluation

and what he calls full noun chunking with different parameters. We will give the figures for the best version of his full noun chunker in Table 4.3. The precision for both ideal PoS-tags and automatic PoS-tag is slightly better than the figures we obtain (ca. 1%), while the recall figures he presents are lower (ca. 4%).

However, the results are not really comparable. He applies his chunker to an ideally tokenized text, which includes the results of named entity recognition. That is, he takes multi-word proper nouns as given, and treats them as single classified tokens. We, however, try to identify named entities and other multi-word units on the basis of rules. The reason is, that we want to present a tool for extraction purposes. If we apply YAC to a large unknown corpus, we cannot expect to have a perfect lexicon including all multi-word units. Thus, we have to find a way to identify them on the basis of rules, and although we are successful in many cases, named entities are nevertheless one of the major error factors in the annotation of YAC. Thus, we can expect to obtain better results, if YAC were applied on a perfect tokenization in the sense of Schiehlen.

Another difference which affects the comparability of the two evaluations is, that Schiehlen evaluates prepositional chunks and noun chunks together. We, however, evaluate PPs and NPs separately.

The example of a comparison between the evaluation figures of YAC and the evaluation figures of Schiehlen’s chunker indicate that it is very difficult to compare different chunking systems even if the underlying theory about the resulting structures is similar.

Skut and Brants (1998) present evaluation figures for their maximum entropy model based on ideal PoS-tags. Precision reaches 93.4% and recall reaches 94.1%. Compared to the results of YAC on ideal PoS-tags, the figures are lower.

Brants (1999) also presents evaluation figures for his Cascaded Markov Model approach. He reports precision and recall for minimal (base) chunks (after 1 layer of annotation), and for maximal chunks (after 9 layers of annotation). Accordingly, the precision for maximal chunks is 91.4% and the recall 84.8%. The figures are based on automatically tagged text. The precision is comparable to that reported by Schiehlen, thus slightly higher than the precision of YAC, however, the recall is considerably lower. Besides, it has to be taken into account that the tool of Brants was both trained and tested on the

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>ideal PoS-tags</td>
<td>97.65</td>
<td>94.06</td>
</tr>
<tr>
<td>automatic PoS-tags</td>
<td>91.11</td>
<td>87.10</td>
</tr>
</tbody>
</table>

Table 4.3: Evaluation figures of the best version of Schiehlen’s full noun chunker
NEGRA corpus, albeit however, on different parts. Thus, the tool was specifically trained for this text.

Müller and Ule (2002) do not present evaluation figures for their tool. According to personal communication, they had similar difficulties as we had in finding a gold standard.
Chapter 5

Applications of YAC-chunked Corpora

5.1 Introduction

As has been pointed out in chapter 2, YAC has been designed especially for use in connection with the extraction of linguistic and lexicographic information. In this chapter, we will present various examples of online extractions performed on the output of YAC. We will show that the off-line annotation provides a useful basis to extract linguistic and lexicographic information from large text corpora. YAC is especially useful as an off-line processor in those cases where the extraction involves complex constructions, such as, e.g., adjectives in predicative(-like) constructions, or adjectives subcategorizing PPs.

In order to extract relevant information, we apply queries to corpora annotated by YAC. The online queries are written in the same formalism as the grammar rules of YAC: CQP. The advantage is that those parts of queries, which are used several times can be annotated, if desired.

The extraction queries can be applied to the corpus in the same way as the grammar use, i.e., using Perl Scripts. The Perl Scripts can be used to filter, and to sort the results, and finally to format the output. Several queries can be processed as a series, and the results can be post-processed, and collected together. We use XML-files to store the results. XSLT style-sheets can then be used to extract the desired data, and transform it in different output formats, e.g., plain text, html, and into a format that can be displayed in a graphical user interface (GUI).

5.2 Sample extractions

We will discuss the following extraction applications of YAC-chunked corpora:
Applications of YAC-chunked Corpora

- subcategorization information
- evidence for scrambling and selectional preferences with respect to distributional variation
- collocation information
- information extraction (named entities)

In addition YAC-annotated corpora could provide a basis for corpus linguistic studies:

- the extraction of multi-word lexemes (e.g., multi-word adverbials)
- morpho-syntactic variations (e.g., singular-plural alternation)
- scrambling phenomena

5.2.1 Subcategorization properties of adjectives

We extracted information about subcategorization properties of adjectives with respect to two different constructions:

1. adjective + PP
2. adjective + verb + (subject/complement) clause

The extracted data can be used not only to determine subcategorization frames, but also selectional preferences and possibly (related) distributional variations resulting from scrambling.

Adjective + PP

We extracted adjective + PP combinations out of a secure context, i.e., a context, where it is clear that the PP belongs to the adjective. We used complex APs embedding PPs as a source. These APs are annotated by YAC under the assumption that the context allows a reliable attachment of the PPs to the adjectival head. We sort out APs with deverbal adjectives as head, which are marked by the feature $\delta$er. Deverbal adjectives are not real adjectives, but verb participles used as adjectives. Thus, information gathered for deverbal adjectives is information about the underlying verb rather than about the adjective.

Yet, although, we extract the adjective + PP pairs out of secure context, some ambiguity remains. Two different types of PPs occur in this context:

1. adverbial modifier PPs, such as in der Tat (in fact) (5.1a)
2. complement PPs, such as stolz auf (proud of) (5.1b)
(5.1)  a.  der [AP [PP in der Tat ] stolze ] Vater
      the in fact proud father
      'the father, who is in fact proud'

      b.  der [AP [PP auf seinen Tochter ] stolze ] Vater
      the of his daughter proud father
      'the father, who is proud of his daughter'

We rarely find locative (5.2a) and temporal (5.2b) adverbial modifiers.

(5.2)  a.  ?der [AP [PP am Bahnhof ] stolze ] Vater
      the at the station proud father
      'the father, who is proud at the station'

      b.  der [AP [PP an jedem Tag ] stolze ] Vater
      the at each day proud father
      'the father, who is proud each day'

The temporal PPs seem to be more likely than the locative ones. As we have
annotated the feature attribute temporal for PPs, it is easy to exclude them.
Besides, we allow only one PP inside the AP. This is due to the fact that if more
than one PP is embedded in the AP, we cannot be sure which PP really belongs
to the adjective. An extracted PP could also belong to the an NP embedded in
one of the other preceding PPs.

Excluding adverbial PPs as in (5.1a) is a bit more problematic. However,
they seem to be more or less fixed expressions of a limited number. The as-
sumption is that subcategorized PoS occur with a limited number of adjectives
only, whereas adverbial PPs of this type can occur with any adjective. Thus, it
should be feasible to differentiate between the two types of PPs. The PPs which
are adverbial modifiers are candidates for multi-words, and can be collected
separately.

The query which is used to extract this kind of information is given below.

MACRO a_pp(0)
(
    <ap>
        /region[advp]*
        <pp>
            []*
            @[._.pp_f not contains "temp"]
        </pp>
    /region[advp]*
    [._.ap_f not contains "vder"
     & _.ap_h != "[A-Z0-9\]+"]*
As can be seen, the query itself can be kept simple, as most of the analysis has already been performed by YAC, when annotating the complex AP structures, the complex AP. Thus, only filter properties have to be specified. For the AP, this means that APs with deverbal heads (\textit{v-d-d}) are excluded. For the embedded PP, this means that PPs with temporal aspect are excluded.

Figure 5.1 shows an example of the results of this query visualized in the LexGUI tool developed by Säuberlich (2002) The GUI interface allows to present the resulting data to lexicographers, who then can decide on the correctness of the proposed properties by checking the respective boxes. Examples are given for each property of a word to help the lexicographer with his decision.

The GUI interface above displays adjectives cooccurring with prepositions. The figures in the main table (left frame) give the frequency of each adjective in such a construction. The list in the right frame gives cooccurrences for one adjective with different prepositions and the respective frequency. At the bottom, examples for each cooccurrence can be displayed. The sample results show the different prepositions cooccurring with the adjective \textit{zuständig} (responsible).

**Adjective + verb + clause**

In the second experiment we extracted adjectives in predicative(-like) constructions subcategorizing a clause. The resulting data can be used to extract different kinds of information:

- subcategorization patterns
- collocations (adjective+verb)
- distributional and selectional preferences
- semantic classes of adjectives

We performed extractions for a variety of syntactic frames each with two different distributions of the clause:

- the clause is in extraposed position at the end of the sentence, following the adjective (5.3–5.12a)
- the clause is in topicalized position at the beginning of the sentence, preceding the verb (5.3–5.12b)

1. adjective + verb + finite clause
5.2 Sample extractions

Figure 5.1: Extraction results for adjective + PP for zuständig
Applications of YAC-chunked Corpora

(5.3) a. Es ist klar, daß . . .
   It is clear, that . . .

   b. Daß . . ., ist klar.
   That . . ., is clear.

   ‘It is clear, that . . .’

2. adjective + reflexive verb + finite clause

(5.4) a. Er ist sich sicher, daß . . .
   He is himself sure, that . . .

   b. Daß . . ., ist er sich sicher.
   That . . ., is he himself sure.

   ‘He is sure, that . . .’

3. adjective + verb + pronominal adverb + finite clause

(5.5) a. Er ist davon überzeugt, daß . . .
   He is of it convinced, that . . .

   b. Daß . . ., davon ist er überzeugt.
   That . . ., of it is he convinced.

   ‘He is convinced, that . . .’

4. adjective + verb + PP[für] + finite clause

(5.6) a. Er hält es für möglich, daß . . .
   He regards it as possible, that . . .

   b. Daß . . ., hält er für möglich.
   That . . ., regards he as possible.

   ‘He regards it as possible, that . . .’

5. adjective + verb + PP[als] + finite clause

(5.7) a. Es gilt als sicher, daß . . .
   It is seen as certain, that . . .

   b. Daß . . ., gilt als sicher.
   That . . ., is seen as certain.

   ‘It is seen as certain, that . . .’

6. adjective + verb + infinite clause

(5.8) a. Es ist unmöglich, ihn zu sehen.
   It is impossible, to see him.

   b. Ihn zu sehen, ist unmöglich.
   To see him, is impossible.

   ‘It is impossible, to see him.’

170
7. adjective+ reflexive verb + infinite clause

(5.9) a. Er ist sich sicher, ihn zu sehen.
   He is himself certain, him to see.
   b. Ihn zu sehen, ist er sich sicher.
   Him to see, is he himself certain.
   'He is certain, to see him.'

8. adjective + verb + pronominal adverb + infinite clause

(5.10) a. Er ist davon überzeugt, ihn zu sehen.
   He is of it convinced, him to see.
   b. Ihn zu sehen, davon ist er überzeugt.
   Him to see, of it is he convinced.
   'He is convinced, to see him.'

9. adjective + verb + PP[für] + infinite clause

(5.11) a. Er hält es für wahrscheinlich, ihn zu sehen.
   He regards it as probable, him to see.
   b. Ihn zu sehen, hält er für wahrscheinlich.
   Him to see, regards he as probable.
   'He regards it as probable, to see him.'

10. adjective + verb + PP[als] + infinite clause

(5.12) a. Es gilt als wahrscheinlich, ihn zu sehen.
   It is seen as probable, him to see.
   b. Ihn zu sehen, gilt als wahrscheinlich.
   Him to see, is seen as probable.
   'It is seen as probable, to see him.'

The extraction query for adjective + verb + finite clause with the finite clause in extraposed position is given below.

MACRO a_v_fincl_ex(0)
(
   ## verbal complex
   <v>
   []
Applications of YAC-chunked Corpora

As can be observed, the query can be kept relatively simple, as the query is composed of annotated structures. The phrasal constituents annotated by YAC are easy to combine into more complex constructions. The query begins with a verbal complex followed by an underspecified number of adjuncts. After this block, there follows an AP, then a comma, and finally a finite subordinate clause. The AP is restricted to APs which are not embedded in NPs but are directly embedded under the sentence or clause level. Deverbal adjectives are filtered out by the Perl-Scripts, and collected in a separate file. The subordinate clause is restricted to clauses introduced by daß/dass (that) or ob (whether). Indirect questions beginning with a wh-word are excluded as they are very difficult to distinguish from (free) relative clauses.

The adjuncts are represented by another macro, which is given below.

MACRO adjct(0) 
(    /phrasal_adjct[]    
|    /nomappo[]    
|    /soappo[]    
|    <cl>    
[_.cl_f contains "rel"]    
[]*  

Adjuncts can be phrasal adjuncts such as AdvPs, PPs, temporal NPs, and interjections, as well as clausal adjuncts such as appositions and relative clauses.

In order to make the fine-grained distinctions described above, this query has to be modified only slightly. In order to find examples for adjective+reflexive verb+finite clause constructions, an obligatory reflexive pronoun is inserted.

```macro
da_v_fincl_ref1_ex(0)
{
  ## verbal complex
  <v>
    []
  </v>
  ## adjuncts and NPs with possible apposition
  /adjct[*]
  <np>
    [...np_f contains "refl"]
    []
  </np>
  /nomappo[]?
  /adjct[*]
  [lemma != "so|derart|zu"]
  ## predicative adjectival phrase
  <ap>
    @[!np
      & word != "so|derart|zu"
      & _ap_h != "[A-Z0-9\-+].*"
    ]
    [word != "so|derart|zu"]
  </ap>
  /comma[]
  ## subordinate clause
  <cl>
    [...cl_h = "dass|daß|ob"]
    []
  </cl>
}
```

In order to find examples for adjective+verb+pronominal adverb+finite clause, the reflexive pronoun is replaced by an obligatory pronominal adverb. For adjective+verb+PP[für]+finite clause, an obligatory PP with the preposition...
Für is inserted, and for adjective+verb+PP[als]+finite clause, an obligatory PP with the preposition als is required.

All these different subcategorization frames are extracted for constructions with the finite verb in topicalized position as well.

MACRO a_v_fincl_top(0)
{
  <s>
    # finite clause
    <cl>
      [_.cl_h = "dass|da$|ob"]
      *
    </cl>
    /comma[]
    # verb
    <v>
      []
    </v>
    # adjuncts
    /adjct[]*
    # adjectival phrase
    <ap>
      @[!np
        & word != "so|derart|zu"
        & _.ap_h != ":[A-Z0-9\+].*"]
        [word != "so|derart|zu"]*
    </ap>
    /region[pp]*
    /dot[]
  }
}

The only difference is that the subordinate clause has to occur before the verbal complex. The rest of the queries remains the same. Topicalized and extrapolosed examples are collected separately, in order to be able to investigate distributional differences with respect to these constructions.

All these different subcategorization frames and distributional differences are collected for the construction with infinite clauses as well.

MACRO a_v_infcl_ex(0)
{
  ### verbal complex
  <v>
    []

174
In this case the subordinate clause is restricted to infinite clauses.

Figure 5.2 shows the summary of all of the collected data for the adjective *klar* (clear). In the right hand column, details about the single phenomena with the respective frequency figures are given.

In order to investigate the results more closely, it is, however, useful to extract parts of the results from the collected data, and display them separately. Table 5.1 shows figures for some adjective+verb cooccurrence figures for adjective+verb+finite verb constructions.

<table>
<thead>
<tr>
<th></th>
<th>fraglich</th>
<th>unklar</th>
<th>klar</th>
<th>offen</th>
<th>möglich</th>
<th>wichtig</th>
<th>deutlich</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>sein</td>
<td>326</td>
<td>320</td>
<td>255</td>
<td>228</td>
<td>160</td>
<td>180</td>
<td>5</td>
<td>1500</td>
</tr>
<tr>
<td>bleiben</td>
<td>34</td>
<td>103</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>177</td>
</tr>
<tr>
<td>machen</td>
<td></td>
<td>41</td>
<td>30</td>
<td></td>
<td>97</td>
<td></td>
<td></td>
<td>168</td>
</tr>
<tr>
<td>werden</td>
<td>3</td>
<td>30</td>
<td>2</td>
<td>2</td>
<td>34</td>
<td></td>
<td></td>
<td>75</td>
</tr>
</tbody>
</table>

Table 5.1: adjective+verb+finite clause

The results allow to make assumptions about the selectional preferences, and possible collocations. If we leave aside the verb *sein* (to be), we can observe a preference of the adjectives *fraglich, unklar,* and *offen* (questionable, unclear, and open) to occur with the verb *bleiben* (to remain), whereas the adjectives *klar, möglich,* and *deutlich* (clear, possible, and another clear) do not occur with *bleiben* but quite often with the verb *machen* (to make).
Applications of YAC-chunked Corpora

Figure 5.2: Extraction results of adjective+verb+clause for klar

176
The same grouping of adjectives can be observed if we look at the distribution of the finite clause displayed in Table 5.2.

<table>
<thead>
<tr>
<th>Adjective</th>
<th>apred_fincl_ex</th>
<th>apred_fincl_top</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>fraglich</td>
<td>91</td>
<td>335</td>
<td>426</td>
</tr>
<tr>
<td>unklar</td>
<td>13</td>
<td>413</td>
<td>426</td>
</tr>
<tr>
<td>klar</td>
<td>221</td>
<td>159</td>
<td>380</td>
</tr>
<tr>
<td>offen</td>
<td>19</td>
<td>266</td>
<td>285</td>
</tr>
<tr>
<td>möglich</td>
<td>207</td>
<td>4</td>
<td>211</td>
</tr>
<tr>
<td>wichtig</td>
<td>192</td>
<td>9</td>
<td>201</td>
</tr>
<tr>
<td>deutlich</td>
<td>139</td>
<td>22</td>
<td>161</td>
</tr>
</tbody>
</table>

Table 5.2: adjective+verb+finite clause sorted by phenomena

In contrast to the other adjectives, *fraglich*, *unklar* and *offen* have a tendency for the finite clause in topicalized position, whereas the other adjectives occur more often with the finite verb in extraposed position. A more detailed discussion of the results can be found in (Kermes and Heid 2003; Heid and Kermes 2002).

The combination of the annotation provided by YAC, with a variety of relatively simple extraction queries allows a fine-grained distinction of the different phenomena. Subcategorization frames, selectional preferences, distributional variations, and possible relations among them can be taken into account. The resulting data gives a good picture of the properties of adjectives in predicative(-like) constructions. More detailed investigation of the results can also provide a basis for semantic clustering.

### 5.2.2 Collocation information

Another application is the extraction of collocation information in the sense of cooccurrences.

**Adjective+noun collocation**

A rather simple experiment extracted adjective+noun cooccurrences from the NEGRA corpus. The reason we choose the NEGRA corpus was that we wanted to evaluate the results, and compare them to the extraction of adjacent pairs, and a window-based methods. The experiment set-up is more closely described in (Evert and Kermes 2003).

The cooccurrence data is extracted from the XML output of YAC by XSLT-style-sheets. In this way, for each noun all embedded adjectives can be collected at once. A few examples of the results are given below.
Applications of YAC-chunked Corpora

absehbar Zeit
eigen Angabe
ersprechend Antrag
ersst Blick
gemischt Gefühl
gut Laune
gut Wille
jed Fall
jed Tag
kommend Woche
kurz Sicht
lauflend Periode
öffentlich Dienst
öffentlich Nahverkehr
persönlich Kontakt
plötzlich Kindstod
politisch Gefangene
rechtsstaatlich Grundsatz
sozial Wohnungsbau
städtisch Kindergarten
steigend Kriminalität
stellvertretend Ortsvorsteher
stellvertretend Vorsitzende
zuständig Ortsbeirat

Precision and recall figures given in Evert and Kermes (2003) are cited in Table 5.3. As (Evert and Kermes 2003) report, adjacent pairs yield the highest

<table>
<thead>
<tr>
<th>candidates from</th>
<th>ideal tags</th>
<th>automatic tags</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>precision</td>
<td>recall</td>
</tr>
<tr>
<td>adjacent pairs</td>
<td>98.47%</td>
<td>90.58%</td>
</tr>
<tr>
<td>window-based</td>
<td>97.14%</td>
<td>96.74%</td>
</tr>
<tr>
<td>YAC chunks</td>
<td>98.16%</td>
<td>97.94%</td>
</tr>
</tbody>
</table>

Table 5.3: Results for Adj+N extraction task

precision for the extraction on ideal tags. However, the recall is considerably lower than for both the window-based, and the chunk approach. Automatic tagging reduces the accuracy by approximately 5%. The chunk-based approach is the least sensitive to tagging errors, and achieves both the best precision and recall figures for this task. Comparing the results of the chunk-based and the window-based approach the difference is not striking, however, a chunk-based approach still leads to the best results.
For a simple extraction task such as adjective+noun collocation, in contrast to the extraction of adjective+verb+clause constructions, a simple window-based approach gives similar results. However, if a chunked corpus is available, even for as simple a task as adjective+noun collocation, a chunk-based approach should be taken into account, as it leads to the best results.

**N+V collocation**

The extraction of noun+verb collocations from dictionaries shows another application possibility of YAC. In this case, the text resource does not consist of full sentences but of fragmentary examples. This poses a problem for full parsers. If possible at all, they would have to be adapted to allow fragments or phrasal structures as well. As YAC uses a bottom-up strategy, fragmentary text can be processed in exactly the same way as full sentences can. From the resulting annotation it is easy to extract the desired noun+verb collocations.

We use two different queries to extract the relevant data. The first extracts evidence for verb+object.

```yacc
MACRO nvobj-coll(0)
(
  <np>
    [_.np_f not contains "temp|ne|pron"
     & !pp
     & _.np_agr contains "Akk:*"]
    []*
  </np>
  /phrasal_adjct[]*
  <vc>
    []*
    @[pos = "VV.*"]
    []*
  </vc>
  </s>
)
```

A few examples of results are given below.

Eindruck machen
Fortschritt machen
Geld machen
Kasse machen
Strich machen
Dampf machen
The queries take the specific character of the text data into account, where the difference between subject and object NP can be determined not only by case, but also by the position of the verb in the fragment. In the case of object NPs, the verb is at the end of the fragment, whereas if the example displays a subject+verb relation, the verbal complex is in the second position of the sentence after the subject NP. Phrasal adjuncts, i.e., PPs, AdvPs, and APs are allowed between both subject and verb, and object and verb, if, however, out of different reasons. In the former case, the phrasal adjuncts are adjuncts of the NP itself, whereas in the latter case, they can be both adjuncts of the NP and adjuncts of the verb.
5.2.3 Extraction of named entities

The annotation of YAC provides a good basis for the extraction of different types of named entities. In the first level of YAC - as described in section 2.4.3 in detail - a variety of special noun chunk rules is applied (see section 3.3.2 for more details) which are used to annotate named entities. Lexical-semantic properties annotated as feature attributes for the chunks can be used to distinguish between the different types of named entities.

As the structures are already annotated, the extraction queries do not have to perform any kind of text analysis but simply have to search for structures with a specific feature attribute. The query below, e.g., searches for street addresses.

MACRO street_address(0)
(
  <nc>
    [._nc_f contains "street"]
    []
  </nc>
)
;
;

As can be observed the query simply looks for a NC structure with the lexical-semantic feature street. A few examples of the results of the query are given below.

Landstraße 102
Arnsburger Straße 24
Radilostraße 17-19
Schloßstraße 35
Brönnerstraße 15
Johannes-Weyer-Str. 1
Giesebrechtstraße 13
Krahmerstraße 6-10
Albert-Schweitzer-Straße 2
Turmstraße 11
Franziusstraße 6
Röderbergweg 82
Bundesstraße 275
Schillerstraße 19
Landstraße 233
Schäfergasse 46
Landstraße 328
Dieselstraße 9
Landstraße 8
Applications of YAC-chunked Corpora

In order to look for another type of named entity, such as, e.g. dates, the query has to be changed only marginally. Instead of the feature street, the NC has to have the feature date.

MACRO date(0)
(
    <nc>
        [._nc_f contains "date"]
        []*
    </nc>
)

A few examples of the results of the query are given below.

27. Oktober 1810
31. Oktober
5. November
Mittwoch, 25. November
26. Dezember
26. Februar
10. 2. 1993
29. Oktober
29. Oktober
7. August
9. September
Juli 1992
31. Oktober
25. Oktober
September 1993
11. Januar 1992
9. Februar 1992
27. Dezember 1991
29. Oktober
März 1992

Unfortunately, an evaluation of the results against the NEGRA corpus was not possible, as the annotation of NEGRA does not include information about lexical-semantic properties. Thus, we decided to make a small manual evaluation with regard to precision for the two named entities: street address and dates. We extracted 100 examples for each query, and checked them for correctness, i.e., whether they really were street addresses and dates. Among the 100 street addresses we extracted 97 where true positives, which corresponds to a precision of 97%. All the 100 dates we extracted where in fact dates corresponding to a precision of 100% in this case. We can only speculate about
recall, which is probably relatively good for dates, as they involve only a limited amount of specific lexemes, which can be listed almost completely. For addresses we can expect that the recall is not as high, as there are a great variety of different street lexemes involved of which we have only a small amount in our lexicon. However, the precision of 97% makes the results useful.

If the query is modified by using the feature sport, the name of sports clubs can be extracted.

Eintracht Frankfurt
AC Parma
Bayern München
FSV Frankfurt
FC Italia
SG Enkheim
TC Marburg
SG Bergen-Enkheim
KC Niederhöchstadt
SKG Frankfurt
BSC Kelsterbach
TuS Kriftel
TV Idstein
TuS Kriftel
SG Sossenheim
TG Höchst
TTV Schmalkalden
SKG Frankfurt
TTV Schmalkalden
SV Darmstadt
SKG Frankfurt
TSV ERFURT
SSV UT ERFURT
SG Wallau/Massenheim

If NC with the feature news are queried, combinations of places or reporters and news agency abbreviates are extracted.

Köln ( dpa )
BÖNN ( dpa )
HAMBURG ( ap )
KLEMENS KINDERMANN ( dpa )
MÜNCHEN ( dpa )
JÖRG FISCHER ( dpa )
WASHINGTON ( dpa )
HAMBURG/MÜNCHEN ( AP )
Applications of YAC-chunked Corpora

WILFRIED MÖMMERT ( dpa )
GELSENKIRCHEN ( dpa )
BERLIN ( dpa )
HUBERT KAHL ( dpa )
HERBERT WINKLER ( dpa )
BRÜSSEL ( AFP )
RÜDIGER EWALD ( dpa )
GÜTTINGER ( AFP )
CHRISTIAN DE VEGT ( dpa )
KLAUS WITTMAN ( AP )
MOSKAU ( AFP )
THOMAS LÜERWEG ( AP )
HUBERT KAHL ( dpa )
BONN ( dpa )
GEROLD BÜCHNER ( AFP )
DAVID CRARY ( AP )
CARSTEN RAVE ( dpa )
BELGRAD ( AP )

Proper names can be extracted using the feature ne.

Philip Glass
Kurt Munkacsi
Rory Johnston
Philips Classics
Uakti
Marco Antonio Guimaraes
Paulo Sergio Santos
Artur Andres
Souza Ramos
Brasilien
Uaktis
Ravel
John Morans
Truman Capote
Lincoln
Manson
Air-Force
Moran
Diktators Mengistu Haile Mariam
Mary Jane V. David
Concerned Artists of Negros
Pröpstin Gundel Neveling Wagener
Meister FV Bad Vilbel

184
Musiker Andreas Heinz Hugo Suberg

It can be assumed that the extraction of proper names will lead to the most noise, as the evaluation of NP structures has shown that proper names are the cause of most false analyses. However, the results are still good enough to provide a good basis for statistical tools, frequency-based approaches or manual inspection.
Chapter 6

Conclusion

We have presented a text analyzing tool for German text (YAC), which is both robust and efficient, and a good basis for the extraction of linguistic and lexicographic information. The goal is to relieve extraction tasks from parts of the linguistic analysis. The main principle behind YAC is that it tries to achieve a maximum of output with a minimum of input. In other words, based on a limited amount of prerequisite knowledge, we want to be able to extract large amounts of information.

We use a relatively small and simple set of rules for the annotation. Nevertheless, we are able to provide a useful basis for extraction processes. The annotated structures are relatively flat in comparison to a full parse. However, in comparison to a classic chunker, the structures are more complex, including complex (recursive) embedding. We try to find a compromise between the goal of providing sufficient structural information, and of annotating reliable structures. The principle behind the underlying chunk definition of YAC is to provide structures which are large enough to provide sufficient information, which are easily combinable into larger structures, and do not involve highly ambiguous attachment decisions (e.g., PP-attachment).

The chunking process is divided into three levels, which all fulfill different tasks. The first level introduces some lexical-semantic properties, annotates base chunks, and annotates chunks with specific internal structures. The second level is the main parsing level, which is applied several times annotating larger and larger chunks with relatively few and simple rules. Structures involving complex (recursive) embedding are gradually built up. The third level, finally, is the finishing level. Dividing the process into different levels helps to avoid undesired interactions among specific rules applied in the first level and the more general rules applied in the second level. Besides, the first level can easily be modified and adapted to specific domains if necessary, without affecting the main chunking.

Simple rules and simple structural annotation means also implementing few theoretical assumptions. This does not make the annotation theory-independent,
Conclusion

but at least less bound to a given linguistic theory. Consequently, YAC can be useful for applications from various theoretical backgrounds.

The rules of YAC are written in CQP, an efficient corpus query language. All rules can be applied as regular queries to a corpus. This allows an interactive rule development and testing. Besides, the results of extraction queries, or parts of extraction queries, can be annotated in the corpus if desired. This can make future extraction easier and faster.

The rules of YAC are applied to the corpus using Perl-Scripts, which are also responsible for the post-processing of results, and for the annotation of the chunks. The post-processing makes it possible to leave rules underspecified as the results can be filtered. The same Perl-Script mechanism can be used to post-process and filter extraction queries.

The good precision figures prove the quality of the chunks annotated by YAC. The precision figures of YAC are almost as high as the figures of other state-of-the-art systems. However, a real comparison is difficult because the single systems do not share the same theoretical background. Besides, Schiehlen (2002) uses a different basis for the annotation. He takes ideal tokens as input, where an ideal token can also be a multi-word unit.

The recall of YAC is as high as its precision, which proves that YAC has a good coverage of phenomena. Other state-of-the-art systems have lower recall figures. A good recall, however, is important for the extraction of phenomena with relatively low frequency such as adjectives in predicative(-like) constructions. In this case we need large amounts of parsed text to be able to extract enough relevant data.

The grammar rules of YAC rely only on lexical information such as PoS-tags, lemma and morpho-syntactic information. The latter is usually ambiguous, and lists all possible options for a word. We do not make use of information about subcategorization frames, selectional preferences, cooccurrences. As we want to extract this kind of information, we cannot depend on it as input to the system. The lexical-semantic information we use is limited to small lists of words. These lists together with simple pattern matching strategies, and certain heuristics can be used to annotate a number of chunks with different lexical-semantic properties.

We do not only annotated extended chunk structures, but enrich the structures additionally with information relevant for extraction. This information includes the head lemma, information about morpho-syntax, and lexical-semantic properties of the head. State-of-the-art chunkers - if at all - provide only some of the relevant information. As has been discussed in chapter 5 this information is important, if not necessary for extraction processes.

Despite the relatively simple and flat annotation, YAC provides a basis for fine-grained distinctions among the extracted data. The additional information allows for a powerful filtering and grouping of the results. We do not only want to extract simple subcategorization frames, but are interested in
selectional preferences, cooccurrences, distributional variations, and relations among the different phenomena. Available systems do not allow to make such a fine-grained distinction.

We presented YAC as a text analyzing tool constructed especially for German. However, it can, in principle, be adapted to any other language which has a concept of phrasal categories. It has already been successfully adapted to Dutch by Kristina Spranger (Spranger 2002; Spranger and Heid 2003) following the same architecture, using a slightly different rule set. This proves that the rules are easy to transfer. The architecture of YAC is especially useful for languages with complex phrasal structures, which allow complex pre-head embedding, such as German, Dutch, and most of the other Germanic languages.

We also plan to conduct an experiment transferring the rule system of YAC to another query language, TIGERSearch, which allows for a better querying of hierarchical structures. Besides, TIGERSearch is based on XML. Thus, we can use XSLT style-sheets for post-processing.

In the future, we want to do more extractions on the output of YAC. We plan to extend the queries for adjectives, and extract other adjective subcategorization frames. There are also more possible queries for adjectives in predicative(-like) constructions, e.g., without clausal complements. The alternation between adjective+verb+PP and adjective+verb+pronominal adverb+clause is also interesting. We also plan to transfer the rules for adjectives to nouns.

Other possible extraction tasks concern multi-word units. Multi-word units pose problems for many Natural Language Processing (NLP) applications. YAC already provides a number of structures, subsuming multi-word units. These structures can be extracted on a relatively simple basis, as has been shown in chapter 5. Based on these annotations, it is possible to create more complex queries to extract more multi-word units. Other multi-word units, which are annotated as normal phrasal categories, can be identified using fine-grained extraction queries and filtering methods.
Zusammenfassung

Immer mehr Texte sind elektronisch verfügbar. Sie enthalten sowohl linguistisches als auch lexicographisches Wissen über Wörter und Wortkombinationen. Die Datenmenge ist jedoch zu groß, um sie manuell verarbeiten zu können. Deshalb brauchen wir (semi-)automatische Werkzeuge, die es erlauben, Text automatisch zu analysieren, und die relevante Information zu extrahieren.

Die Textanalyse kann sowohl off-line als auch online erfolgen. Eine reine online Verarbeitung ist für Extraktionszwecke jedoch nur wenig hilfreich. Sie produziert zu viel Rauschen und stellt zu wenig Information zur Verfügung. Die Frage ist daher, wieviel der Analyse off-line und wieviel online ausgeführt werden soll.

Die Hypothese ist: je detaillierter die off-line Analyse, desto besser und schneller ist die online Extraktion. Jedoch, je komplexer die Annotation, desto komplexer die Grammatik, desto schwieriger und langwieriger die Entwicklung der Grammatik und desto langsamer ist der Extraktionsprozess.

6.1 Voraussetzungen

Die Frage ist, welche Analysetiefe die Voraussetzungen für Extraktionsanfragen am Besten erfüllt. Für uns sind die zentralen Voraussetzungen für ein Textanalysetool die folgenden: (i) das Tool muß in der Lage sein unbegrenzten Text zu verarbeiten. Es sollte also kein Limit für Korpusgröße geben, sowohl große als auch kleine Textmengen sollten verarbeitet werden können. Es sollten sowohl ganze Sätze als auch fragmentarischer Text analysiert werden können. Das System sollte auch nicht auf eine bestimmte Domäne beschränkt sein, d.h., es sollte möglichst auf alle Textarten anwendbar sein. Domänspezifische Regeln sollten leicht einfügbare sein. (ii) Lücken in der Grammatik sollten nicht automatisch zum Scheitern der gesamten Analyse führen. (iii) Eine manuelle Korrektur sollte nicht nötig sein, weil eine solche auf großen Textmengen nicht realisierbar ist. (iv) Das System sollte klar definierte Schnittstellen haben, auf die die Extraktion aufsetzen kann. Die Annotation sollte sich außerdem linguistischer Standards bedienen. Es sollte eine Dokumentation geben, in der aufgeführt wird, was und wie es annotiert ist. Die Annotation sollte für spätere
Zusammenfassung

Anwendungen gut zu verarbeiten sein.

Die Annotation selbst sollte folgende Informationen bereitstellen: (i) das Kopflemma, um den Kopf von lexikalischen Einträgen zu bestimmen, (ii) morpho-syntaktische Information, um die grammatische Funktion der Chunks zu bestimmen, und um eventuell zusätzliche Information über die Einträge zu sammeln (wie z.B. singular-plural Alternationen). (iii) Lexikalisch-semantische Information wie z.B. temporalen Aspekt und (v) eine hierarchische Repäsentation.


Wie Kübler und Hinrichs (2001) es formulieren: während das Chunken bisher auf das Identifizieren von partiellen Konstituentenstrukturen auf der Ebene von einzelnen Chunks spezialisiert ist, wird der Frage, wie die partiellen Analysen zu größeren Ausdrücken zusammengefügt werden können kaum oder gar kein Aufmerksamkeit geschenkt.

6.2 YAC - Ein rekursiver Chunker für unbegrenzten Deutschen Text

YAC is ein vollautomatischer rekursiver Chunker für unbegrenzten Deutschen Text. Er basiert auf einer Grammatik, die aus regulären Ausdrücken besteht. Die Regeln sind in der CQP Anfragesprache (Christ et al. 1999) geschrieben, die Teil der IMS Workbench (CWB) ist.


YAC ist in mancher Hinsicht ein typischer Chunker: (i) er ist robust, d.h. er kann unbegrenzten Text verarbeiten, (ii) er arbeitet vollautomatisch, (iii) er stellt keine volle Analyse sondern eine Teilanalyse zur Verfügung, (iv) er fällt keine hochgradig ambigen Attachmententscheidungen.

YAC unterscheidet sich jedoch auch von anderen Chunkern: (i) er erweitert die klassische Chunkdefinition von Abney, (ii) er stellt zusätzliche Informationen zu den annotierten Chunks bereit.
Die Chunkdefiniton von Abney (1996a) als ein “non-recursive core of an intra-clausal constituent, extending from the beginning of the constituent to its head” wird durch zwei Aspekte erweitert: (i) rekursive Einbettung, (ii) Einbettung nach dem Kopf.

Die von YAC annotierten Chunks sind immer noch teilsatzinterne Konstituenten. Sie sind jedoch nicht mehr nicht-rekursiv sondern beinhalten rekursive Einbettung vor (pre-head) (6.1a) sowohl als auch nach (post-head) (6.1b) der Kopfposition.

(6.1) a. \[NP\] die kleinen, über \[NP\] die Köpfe der Apostel \] gesetzten Flammen \]

b. \[NP\] die Köpfe \[NP\] der Apostel \]

c. \[NP\] Jahre \[AdvP\] später \]


Es gibt jedoch gewisse Einschränkungen hinsichtlich der post-head Modifikatoren. YAC annotiert nur Strukturen, die nicht ambig sind. Deshalb werden hochgradig ambige Attachmententscheidungen, wie z.B. PP-attachment, nicht gefällt. Um diese Ambiguitäten aufzulösen, wäre umfangreiches lexikalisches, linguistisches, und kontextuelles Wissen, in einigen Fällen sogar Weltwissen erforderlich.

Die Chunkdefinition von Abney wird daher wie folgt ergänzt und neu formuliert:

(6.2) A chunk is a continuous part of an intra-clausal constituent including recursion, pre-head as well as post-head modifiers, but no PP-attachment, or sentential elements.

Zusätzlich werden die Chunks mit dem Kopflemma, morpho-syntaktischer Information, sowie einigen lexikalisch-semantischen Eigenschaften angereichert.

### 6.2.1 Annotierte Strukturen

Die Strukturen, die YAC annotiert umfassen folgende lexikalische Phrasenkategorien: (i) Adverbialephrasen (AdvP), (ii) Adjectivphrasen (AP), (ii) Nominalphrasen (NP), (iv) Prepositionalphrasen (PP), (v) Verbalkomplexe (VC), (vi) einzelne Verben (V), (vii) Teilsätze (CL).
Zusammenfassung

**Adverbialphrasen** sind relativ einfache Konstruktionen, die aus einem Adverb und einem optionalen Partikel bestehen. In einigen Kontexten kann der Kopf durch ein weiteres Adverb modifiziert werden. Beispiel (6.3) zeigt mögliche AdvPs.

(6.3) vielleicht; zu bald

**Adjectivphrasen** können als einfache oder als komplexe Phrasenstrukturen realisiert werden. Die einfachste AP besteht aus einem Adjektiv als Kopf (6.4a). Etwas komplexere APs umfassen zusätzlich AdvPs (6.4b). Andere APs können spezielle Konstruktionen (6.4c), oder komplexe Einbettungen (6.4d) beinhalten, die vom lexikalischen Kopf gefordert werden.

(6.4)

a. möglich
b. schreiend lila
c. rund zwei Meter hohe
d. über die Köpfe der Apostel gesetzten

**Nominalphrasen** umfassen einfache Konstruktionen, die aus einem einzelnen Nomen oder einem Pronomen bestehen (6.5a). Außerdem umfassen NPs auch spezifische Konstruktionen wie in (6.5b-c), sowie komplexe Konstruktionen mit Rekursion in pre-head Position (6.5d).

(6.5)

a. Oktober; er
b. 4,9 Milliarden Dollar
c. "Frankensteins Fluch"
d. kleine, über die Köpfe der Apostel gesetzte Flammen

**Prepositionalphrasen** umfassen Pronominaladverbien wie in (6.6a), und komplexere Strukturen, die koordinierte NPs (6.6b), sowohl als auch komplexe NP (6.6c) einbetten können.

(6.6)

a. davon
b. zwischen Basel und St. Moritz
c. mit kleinen über die Köpfe der Apostel gesetzten Flammen

**Verbale Komplexe** sind einfache Strukturen, die aneinanderhängende verbale Elemente umfassen (6.7).

(6.7)

a. gemunkelt
b. muß gerechnet werden
c. zu bekommen
6.2 YAC - Ein rekursiver Chunker für unbegrenzten Deutschen Text

Teilsätze werden durch Aneinandereihung der annotierten lexikalischen Phrasen gebildet. Einerseits beweisen die Teilsatzregeln, daß dieChunks tatsächlich auf einfache Weise zu größeren Strukturen zusammengebaut werden können, andererseits werden die Teilsätze selbst zum Extrahieren linguistischer Evidenz benutzt.

6.2.2 Feature Annotation

Wir annotieren auch Feature Attribute, die einige Eigenschaften der Chunks näher bestimmen. Weil jede Chunkkategorie andere Charakteristika hat, haben wir für jede Chunkkategorie ein eigenes Annotationsschema. Tabelle 6.1 gibt einen Überblick über die Feature Annotation aller Chunkkategorien.

<table>
<thead>
<tr>
<th>Featurewerte</th>
<th>AdvP</th>
<th>AP</th>
<th>NP</th>
<th>PP</th>
<th>VC</th>
<th>CL</th>
</tr>
</thead>
<tbody>
<tr>
<td>lexikalisch-semantische Eigenschaften (f)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Kopflemma (h)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Agreement Information (agr)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>letzter Buchstabe des Kopflemmas (suff)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verbales Kopflemma (v1em)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: Feature Annotation der einzelnen Chunkkategorien

Lexikalisch-semantische Eigenschaften sind sowohl für die Textanalyse als auch für zukünftige Anwendungen wichtig. Die Eigenschaften können Auslöser für eine spezielle interne Struktur, Funktion, oder die Verwendung der Chunks sein. Einige der Eigenschaften sind im Korpus selbst enthalten, d.h. sie können direkt aus dem Korpus abgeleitet werden, z.B. von PoS-Tags oder Textmarkierungen. So können Eigennamen u.a. vom PoS-Tag NE, das für Eigennamen steht, abgeleitet werden (6.8).

(6.8) \[NP \text{ Johann Sebastian Bach} \]
\[NE \quad NE \quad NE\]

Textmarkierungen wie etwa Anführungszeichen oder Klammern zeigen den speziellen Charakter (Eigennamen oder möglicher Modifizierer) einer Struktur an (6.9). Sie können als sicherer Kontext benutzt werden, um Einschränkungen, denen Chunks unterliegen, zu lockern.

(6.9) "Wilhelm Meisters Lehrjahre"

Eine andere Möglichkeit Eigenschaften von Chunks herzuleiten, ist durch den Chunkingprozess selbst. In diesem Fall, bilden spezifische Einbettungen die Eigenschaften der einbettenden Chunks. Komplexe APs die PPs (6.11a) und NPs (6.11b) einbetten werden so mit dem jeweiligen Feature markiert: pp für PP-Einbettung und np für NP-Einbettung.

(6.11) a. \[ AP [ PP \text{ über die Köpfe der Apostel } ] \text{ gesetzten } \]

b. \[ AP [ NP \text{ der "Inkatha"-Partei } ] \text{ angehörenden } \]

### 6.2.3 Chunkingprozess


Die Tatsache, daß die Grammatikregeln in einer Anfragesprache geschrieben sind, erlaubt eine interaktive Grammatikentwicklung. Der Textanalyseprozess kann zu jedem Zeitpunkt angehalten werden, um neue oder geänderte Regeln auf dem aktuellen Stand des Korpus in einer interaktiven CQP-Sitzung zu testen.

Derselbe Formalismus, der für die Grammatikregeln benutzt wird, kann für eine interaktive Anfrage benutzt werden. Templates, die Strukturen abdecken, die sich als relevant herausstellen, können auf einfache Art und Weise in den Analyseprozess eingebunden werden, wenn das gewünscht wird. In diesem Fall wird die Regel einfach als Grammatikregel mit den anderen Regeln angewendet, und die Ergebnisse werden im Korpus annotiert.


Der Chunkprozess ist in drei verschiedene Stufen unterteilt, die verschiedenen Aufgaben erfüllen:

6.3 Evaluierung

- Zweite Stufe (second level): (i) Hauptanalysestufe, (ii) iterative Anwendung von allgemeinen Phrasenstrukturregeln, um komplexe Phrasenstrukturen aufzubauen

- Dritte Stufe (third level): Abschlußstufe

Abbildung 6.1 zeigt die Architektur des Chunkprozesses.

Abbildung 6.1 zeigt die Architektur des Chunkprozesses.

Das Annotieren von Basischunks mit spezifischer interner Struktur und lexikalisch-semantische Information auf der ersten Stufe bringt zahlreiche Vorteile mit sich: (i) Die speziellen Regeln der ersten Stufe interagieren nicht mit den allgemeineren Regeln der Hauptanalyse, (ii) die Regeln, die keine komplexen (rekursiven) Einbettungen beinhalten, müssen nur einmal angewendet werden, (iii) zusätzliche Regeln, die notwendig sind um spezifische Phänomene oder spezielle Textdomänen abzudecken, können auf einfache Art und Weise eingebunden werden, ohne daß die Hauptanalyse davon betroffen ist, (iv) die Regeln der Hauptanalysestufe können relativ einfach und allgemein gehalten werden, weil die meisten Spezialfälle bereits abgedeckt sind, (v) nur eine relativ kleine Menge von Regeln sind für die Hauptanalyse erforderlich.

6.3 Evaluierung

Wir haben zwei verschiedene Evaluierungen durchgeführt: eine für NPs auf der Basis des NEGRA Korpus (Skut et al. 1998), und eine für APs, NPs, PPs und VCs auf der Basis eines manuell erstellten Referenzkorpus. Für den ersten Ansatz mußten wir NP Chunks aus dem NEGRA Korpus heraussschneiden. Dazu benutzten wir das TigerXML-Format (Mengel and Lezius 2000) und
Zusammenfassung

XSLT Style-Sheets. Leider ist die syntaktische Annotation der NEGRA Baumbank (Skut et al. 1997), die alle Projektionen, die für das Ableiten der Konstituentenstruktur nicht unbedingt erforderlich sind, weggelassen, für eine automatische Ableitung einer Chunkreferenz nicht geeignet. Deshalb war es uns nicht möglich, einen sauberen und korrekten Goldstandard zu extrahieren.

Wir haben einen weiteren Ansatz auf einem manuell erstellen Referenzkorpus ausgeführt, um zu sehen, inwieweit die Zahlen vom Rauschen in der NEGRA Referenz beeinflusst wurden, und um ein besseres Bild der tatsächlichen Leistungsfähigkeit von YAC zu bekommen. Der Text für des manuell erstellten Referenzkorpus besteht aus 400 Sätzen, die zufällig aus dem NEGRA Korpus herausgezogen wurden.

Wir haben die Leistungsfähigkeit von YAC auf manuell annotierten PoS-Tags (idealen Tags), und auf automatischen annotierten PoS-Tags (automatischen Tags) evaluiert. In beiden Fällen ist die morpho-syntaktische Information nicht manuell disambiguiert, sondern bleibt ambig. Bei der Evaluierung auf der Basis der manuell erstellten Referenz haben wir außerdem Werte für alle Chunks und Werte für maximale Chunks ermittelt.

6.3.1 Evaluierungsergebnisse

Die Evaluierung von YAC auf idealen Tags auf der Basis des NEGRA Korpus ergab eine Präzision von 88,21% und einen Recall von 90,03%. Die gleiche Evaluierung auf automatischen Tags ergab eine Präzision von 82,48% und einen Recall von 86,11%.


<table>
<thead>
<tr>
<th></th>
<th>alle Chunks</th>
<th>maximale Chunks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Präzision</td>
<td>Recall</td>
</tr>
<tr>
<td>NP</td>
<td>96,36</td>
<td>96,51</td>
</tr>
<tr>
<td>PP</td>
<td>98,08</td>
<td>96,51</td>
</tr>
<tr>
<td>AP</td>
<td>96,39</td>
<td>97,50</td>
</tr>
<tr>
<td>VC</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 6.2: Evaluierungswerte von YAC auf idealen Tags

Tags sind in Tabelle 6.3 aufgeführt.

198
### 6.4 Anwendungen


Wir haben, z.B., Informationen über Subkategorisierungseigenschaften von Adjektiven aus zwei verschiedenen Konstruktionen extrahiert:

1. Adjektiv + PP
2. Adjektiv + Verb + Teilsatz

<table>
<thead>
<tr>
<th></th>
<th>alle Chunks</th>
<th>maximale Chunks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Präzision</td>
<td>Recall</td>
</tr>
<tr>
<td>NP</td>
<td>89,93</td>
<td>91,67</td>
</tr>
<tr>
<td>PP</td>
<td>94,05</td>
<td>89,67</td>
</tr>
<tr>
<td>AP</td>
<td>84,24</td>
<td>89,25</td>
</tr>
<tr>
<td>VC</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 6.3: Evaluierungswerte von YAC auf automatisch annotierten Tags


Zusammenfassung


Wir haben Adjektiv + PP Kombinationen aus sicheren Kontexten extrahiert, d.h. aus einem Kontext, bei dem die Zugehörigkeit der PP zum Adjektiv eindeutig ist. Wir haben dafür komplexe APs, die PPs einbetten, benutzt. APs mit deverbalen Adjektiven als Kopf, die mit dem Feature derer markiert sind, werden aussortiert. Diese Adjektive sind keine Adjektive im eigentlichen Sinn, sondern vielmehr Verbpartizipien, die wie Adjektive verwendet werden.

Im zweiten Experiment haben wir Adjektive in prädikativ(-artigen) Konstruktionen, die einen Teilsatz subkategorisieren, extrahiert. Die Daten die aus dieser Extraktion hervorgehen, können dazu benutzt werden folgende Informationen abzuleiten:

- Subkategorisierungsmuster
- Kollokationen (Adjektiv + Verb)
- distributionelle und Selektionspräferenzen
- Semantische Klassen

Die Kombination der von YAC bereitgestellten Annotation und einer Vielzahl relativ einfacher Extraktionsanfragen, erlaubt es differenzierte Unterscheidungen zu machen, die ein gutes Bild der Eigenschaften und Subkategorisierungsmuster von Adjektiven geben. Eine derartig detaillierte Analyse kann auch die Basis für semantisches Clustern sein.

Summary

6.5 Introduction

More and more text corpora are available electronically. They contain information about linguistic and lexicographic properties of words, and word combinations. The amount of data is too large to extract the information manually. Thus, we need means for a (semi-)automatic processing, we need to analyse the text to be able to extract the relevant information.

The text analysis can be performed off-line by a text analysing tool, or on-line by task specific queries. An online analysis leads to too much noise, and too little information to produce useful results. Thus, the workload has to be divided between an off-line and an online stage of analysis. The question is, how much of the text analysis should be performed off-line, and how much online.

The hypothesis is that the better and more detailed the off-line annotation, the better and faster the on-line extraction. However, the more detailed the off-line annotation, the more complex the grammar, the more time consuming and difficult the grammar development, and the slower the parsing process.

6.6 Requirements

The question is which depth of analysis fits best the needs of queries for the extraction of linguistic and lexicographic information. We understand the crucial requirements for a useful tool to be the following: (i) It has to work on unrestricted text. There should be no limitation to corpus size, i.e., it should be able to deal with small as well as large corpora. It should be able to parse complete sentences as well as fragmentary text. The system should not be domain specific, i.e., it should work basically on any text type. Additional domain specific rules should be easy to incorporate in the grammar. (ii) Shortcomings in the grammar should not lead to a complete failure to parse. (iii) No manual checking should be required as this is not feasible for large quantities of text. (iv) The system should provide clearly defined and well documented interfaces, where the extraction processes can attach. The annotation should make use of
linguistic standards. There should be a documentation on what is annotated, and how it is annotated. The annotation should be easy to process for further application.

There are also certain requirements for the corpus annotation in order to provide a useful basis for extractions, besides the information on token level: (i) the head lemma of annotated structures to determine the lemma of the lexical entry, (ii) morpho-syntactic information, to determine the grammatical function of the structure and to extract additional aspects of potential lexical entries, e.g., singular plural alternations for nouns, (iii) lexical or semantic information, e.g., temporal aspect, (iv) information about certain embeddings, text markers, or construction types, (v) hierarchical representations.

The best, richest and most complex, and usually most reliable basis (especially if manual checking is involved) is provided by a full parse. However, full parsers often lack robustness and/or provide ambiguous output except for (unreliable) statistical disambiguation. Besides, they often have a slow parsing speed, which makes working with large corpora tedious. A chunk analysis, being robust, fast and considerably reliable seems a good alternative for large scale corpus linguistic work. At the moment, chunking is the only technically feasible approach. Yet, the question remains: can a chunker provide the relevant information listed above?

As Kübler and Hinrichs (Kübler and Hinrichs 2001) have pointed out, while chunking approaches have “focused on the recognition of partial constituent structures at the level of individual chunks [. . . ], little or no attention has been paid to the question of how such partial analyses can be combined into larger structures for complete utterances.”

6.7 YAC - A recursive chunker for unrestricted German text

YAC is a fully automatic recursive chunker for unrestricted German text. It is based on a symbolic regular expression grammar written in the CQP query language (Christ et al. 1999) which is part of the IMS Corpus Workbench¹. The chunker works on a corpus which is tokenized and part-of-speech tagged using the STTS-tagset (Schiller et al. 1999). For tokenization and PoS-tagging the TreeTagger² (Schmid 1994; Schmid 1995) is used. The German grammar additionally requires lemma and agreement information on token level, which is annotated using the IMSLex morphology (Lezius et al. 2000).

In some respects YAC is a typical chunker: (i) it is robust, i.e., it works on unrestricted text, (ii) it works fully automatically, (iii) it does not provide a

---

¹see http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench
²see http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html
full but only a partial analysis of text, (iv) it does not make highly ambiguous attachment decisions.

But YAC is also different from other state-of-the-art chunkers: (i) It extends the classic chunk definition of Abney. (ii) It provides additional information about the annotated chunks.

### 6.7.1 Chunk definition

The classic chunk definition of Abney as a “non-recursive core of an intra-clausal constituent, extending from the beginning of the constituent to its head” (Abney 1996a) is extended by two main aspects: (i) recursive embedding, (ii) post-head embedding.

The chunks annotated by YAC are still intra-clausal constituents, however, they are no longer non-recursive but include recursive embedding in pre-head (6.12a) as well as in post-head position (6.12b). Post-head recursion implies that the chunk does not end with the head but may have modifiers in post-head position. The English of-phrases are often realized as post-head nominal genitive modifiers as is the case in (6.12b). Thus, it is useful if not necessary to attach them. Post-head modifiers can but do not necessarily have to have the same category as the chunk itself. In (6.12c), e.g., an adverbial phrase is embedded in an NP in post-head position.

(6.12) a. \( \text{[NP die kleinen, über \[NP die Köpfe der Apostle \] gesetzten Flammen]} \)  
   'the small, above the heads of the apostles set flames'

b. \( \text{[NP die Köpfe [NP der Apostle]]} \)  
   'the heads of the apostles'

c. \( \text{[NP Jahre [AdvP später]]} \)  
   'year later'

There are, however, certain limitations with respect to post-head modifiers. YAC annotates only non-ambiguous constructions. Consequently, highly ambiguous attachment decisions, such as PP-attachment, are not made. Solving these ambiguities requires comprehensive lexical, linguistic, and context information, and in some cases world knowledge.

The chunk definition of Abney is extended and reformulated as follows:

(6.13) A chunk is a continuous part of an intra-clausal constituent including recursion, pre-head as well as post-head modifiers, but no PP-attachment, or sentential elements.

The chunks are additionally enriched with head lemma and morpho-syntactic information as well as certain lexical-semantic properties.
Summary

6.7.2 Annotated structures

The structures annotated by YAC comprise the following lexical phrase categories: (i) adverbial phrases (AdvP), (ii) adjectival phrases (AP), (iii) noun phrases (NP), (iv) prepositional phrases (PP), (v) verbal complexes (VC), (vi) single verbs (V), (vii) subordinate clauses (CL).

Adverbial phrases Adverbial phrases are relatively simple constructions consisting of an adverb and an optional particle. In certain contexts, an optional adverbial modifier is allowed. Examples of possible AdvPs are given in (6.14).

(6.14) vielleicht (perhaps); zu bald (to soon)

Adjectival phrases can be simple as well as complex phrase structures. The most simple APs consist solely of the adjective head as in (6.15a). Slightly more complex are APs embedding adverbial structures as in (6.15b). Other APs can comprise specific constructions triggered by a certain lexical head as in (6.15c), and complex embeddings as in (6.15d).

(6.15) a. möglich (possible)
   b. schreiend lila
      screamingly purple
   c. rund zwei Meter hohe
      around two meters high
   d. über die Köpfe der Apostel gesetzten
      above the heads of the apostles set
      set above the heads of the apostles

Noun phrases range from simple constructions consisting of a single noun or pronouns as in (6.16a), specific constructions as in (6.16b–c) to complex constructions involving recursion in pre-head positions as in (6.16d).

(6.16) a. Oktober (October); er (he)
   b. 4,9 Milliarden Dollar
      4.9 billion dollar
   c. "Frankenstein’s Fluch" 
      Frankenstein’s curse
   d. kleine, über die Köpfe der Apostel gesetzten Flammen
      small, above the heads of the apostles set flames
      small flames set above the heads of the apostles
Prepositional phrases comprise pronominal adverbs as in (6.17a) and more complex structures embedding coordinated NPs as in (6.17b) as well as complex NPs as in (6.17b–c).

(6.17) a. davon (thereof)
   b. zwischen Basel und St. Moritz
   between Basel and St. Moritz
   c. mit kleinen über die Köpfe der Apostel gesetzten Flammen
   with small above the heads of the apostles set flames
   with small flames set above the heads of the apostles

Verbal complexes are simple structures comprising of adjacent verbal elements as in (6.18).

(6.18) a. gemunkelt (rumored)
   b. muß gerechnet werden
   has counted to be
   has to be counted
   c. zu bekommen
   to get

Subordinate Clauses are assembled using the annotated lexical phrases. On the one hand, the rules used to build the clauses prove that the chunks annotated by YAC are easily combined to larger clausal constructions. On the other hand, the clauses themselves useful are for the extraction of linguistic evidence.

6.7.3 Feature annotation

We additionally annotate feature attributes specifying certain properties of chunks. As each category has different characteristics, we have different annotation schemes for each category. Table 6.4 gives an overview of the feature annotation for all chunk categories.

Lexical-semantic and structural properties are important for parsing as well as for further applications. The properties can be triggers for specific internal structures, functions, and the usage of chunks. Some of the properties are inherent in the corpus itself, i.e., they can be determined from the information already present in the corpus: (i) PoS-tags, (ii) text markers. Named entities, e.g., can be derived from the PoS-tag NE for proper noun (6.19).

(6.19) [NP Johann Sebastian Bach]
   NE      NE      NE
Summary

<table>
<thead>
<tr>
<th>feature value</th>
<th>AdvP</th>
<th>AP</th>
<th>NP</th>
<th>PP</th>
<th>VC</th>
<th>CL</th>
</tr>
</thead>
<tbody>
<tr>
<td>lexical-semantic properties (f)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>head lemma (h)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>agreement information (agr)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>last letter of head lemma (suff)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>verbal head lemma (v1em)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Table 6.4: Feature annotation for all chunk categories

Text markers such as quotation marks, parentheses, and brackets indicate the special character of a chunk (e.g. as named entity or possible modifier) (6.20), and can function as a secure context in which the restrictions on the chunks are relaxed.

(6.20) "Wilhelm Meisters Lehrjahre"

Other properties are determined by external knowledge sources, such as lexicons and ontologies. Local adverbs (6.21), e.g., are identified according to manually prepared word lists.

(6.21) hier (here); dort (there)

Another possibility to derive properties is from the chunking process itself. In this case, specific embeddings are marked as properties of the embedding chunk. Complex APs embedding PPs (6.22a) and NPs (6.22b) are marked by a respective feature indicating the embedded structure.

(6.22) a. \([AP_{PP} \text{ über } \text{die Köpfe der Apostel }] \text{ gesetzten }\]

   above the heads of the apostles set

   ‘set above the heads of the apostles’

b. \([AP_{NP} \text{ der } \text{"Inkatha"-Partei }] \text{ angehörenden }\]

   to the Inkatha-Partei belonging

   ‘belonging to the Inkatha-Partei’

6.7.4 Chunking process

The grammar rules of YAC are written in an efficient query language (CQP). The results of the rules are post-processed by separate Perl-scripts. Powerful compression algorithms allow to work with large corpora. Even complex queries can be efficiently evaluated and processed. It is possible to work with corpora of 200-300 million tokens. The query language is modular, i.e., it allows to split complex rules into different blocks. The fact that the grammar rules are
written in a query language allows an interactive development and testing of the rules. The parsing process can be interrupted at any time in order to test new or changed rules on the current state of the corpus in an interactive CQP session.

The same formalism used for the grammar rules can be used for interactive querying of the final results. Templates covering structures which are found to be relevant can be easily included in the parsing process if desired. In this case, the query is simply taken as a grammar rule, is applied together with the other rules, and the result is annotated in the corpus.

The use of Perl-scripts as a framework for the rule system allows a powerful post-processing of the results of the grammar rules. Results can be filtered, changed, and sorted, if necessary. This allows grammar rules to be left underspecified. Lexical information can easily be included without multiplying the grammar size unnecessarily. Different output formats can be provided, and hierarchical structures can be built.

The chunking process is divided into three levels, which serve different purposes:

- First Level: (i) annotates base-chunks (in the sense of Abney), (ii) annotates chunks with a specific internal structure, (iii) introduces lexical-semantic properties
- Second Level: (i) main parsing level, (ii) iterative application of general phrase structure rules to build complex chunk structures
- Third Level: finishing level

Figure (2.6) illustrates the architecture of the chunking process of YAC.
Summary

There are several advantages of annotating base-chunks with specific internal structures and introducing lexical and semantic information in the first level: (i) the specific rules do not interact with the main parsing rules, (ii) the rules for chunks which do not involve complex (recursive) embedding have to be applied only once, (iii) the additional rules which are necessary to cover specific phenomena or specialized text domains can be included easily without affecting the main parsing process, (iv) the rules of the main parsing process can be kept relatively simple and general, as most special cases are already covered, (v) only a relatively small number of "general" rules is needed for the main parsing process.

6.8 Evaluation

We performed two different evaluations: one based on the NEGRA corpus (Skut et al. 1998) for NPs, and the other on a manually built reference corpus for NPs, PPs, APs, and VCs. For the first approach we had to cut out the NP chunk reference from the NEGRA corpus using the TigerXML format (Mengel and Lezius 2000) and XSLT style-sheets. Unfortunately, the syntactic annotation scheme of the NEGRA treebank (Skut et al. 1997), which omits all projections that are not strictly necessary to determine the constituent structure of a sentence, is not very well suited for an automatic derivation of a chunk reference. Thus, we were not able to extract a clean and correct gold standard.

We performed another evaluation on a manually built reference to see how much the figures were influenced by the noise in the NEGRA reference, and to get a better picture about the real performance of YAC. The text for the manually built reference are 400 sentences taken from the NEGRA corpus at random.

We evaluated the performance of YAC on manually disambiguated PoS-tags (ideal tags), and on PoS-tags provided by a statistical tagger (here: the TreeTagger (TreeTagger; Schmid 1994; Schmid 1995)) (automatic tags). In all cases, the morpho-syntactic information is not manually disambiguated but left ambiguous. For the evaluation against the manual reference we additionally distinguished between the figures for all chunks, and for maximal chunks.

6.8.1 Evaluation results

The evaluation of YAC on ideal PoS-tag against the NEGRA corpus leads to a precision of 88.21% with a recall of 90.03%. The same evaluation on automatic PoS-tags leads to a precision of 82.48% and a recall of 86.11%.

The extraction of an NP reference from the NEGRA corpus was not really successful. The NEGRA reference cannot be seen as a gold standard for YAC. More reliable are the figures gained from the manual reference. The evaluation
figures of YAC against the manual reference on ideal tags are given in Table 6.5. The corresponding evaluation on automatic PoS-tags is given in Table 6.6.

<table>
<thead>
<tr>
<th></th>
<th>all chunks</th>
<th>maximal chunks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>precision</td>
<td>recall</td>
</tr>
<tr>
<td>NP</td>
<td>96.36</td>
<td>96.51</td>
</tr>
<tr>
<td>PP</td>
<td>98.08</td>
<td>96.51</td>
</tr>
<tr>
<td>AP</td>
<td>96.39</td>
<td>97.50</td>
</tr>
<tr>
<td>VC</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 6.5: Evaluation figures of YAC on ideal PoS-tags

<table>
<thead>
<tr>
<th></th>
<th>all chunks</th>
<th>maximal chunks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>precision</td>
<td>recall</td>
</tr>
<tr>
<td>NP</td>
<td>89.93</td>
<td>91.67</td>
</tr>
<tr>
<td>PP</td>
<td>94.05</td>
<td>89.67</td>
</tr>
<tr>
<td>AP</td>
<td>84.24</td>
<td>89.25</td>
</tr>
<tr>
<td>VC</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 6.6: Evaluation figures of YAC on automatic PoS-tags

The good precision figures prove the quality of the chunks annotated by YAC. The precision figure of YAC more or less equal those of other state-of-the-art systems. Schiehlen (2002) reports a precision of 97.65% and a recall of 94.06% on ideal PoS-tags, and a precision of 91.11% and a recall of 87.10% on automatic PoS-tags for maximal full noun chunks. Skut and Brants (1998) report a precision of 93.4% and a recall of 94.1% for their maximum entropy model on ideal PoS-tags. Brants (1999) reports a precision of 91.4% and a recall of 84.8% for maximal full noun chunks on automatic PoS-tags. However, a real comparison is difficult because the single systems do not share the same theoretical background. Besides, Schiehlen (2002) uses a different basis for the annotation. He takes ideal tokens as input, where an ideal token can also be a multi-word unit.

Recall and precision figures of YAC are almost equally high, which proves that YAC has a good coverage of phenomena. Other state-of-the-art systems have lower recall figures. A good recall, however, is important for the extraction of phenomena with relatively low frequency figures such as adjectives in predicative(-like) constructions. In this case we need large amounts of parsed text to be able to extract enough relevant data.
6.9 Application

YAC has been designed especially for the use in connection with the extraction of linguistic and lexicographic information. In order to extract relevant information, we apply queries to corpora annotated by YAC. The queries are written in the same formalism as the grammar rules of YAC. Perl-Scripts can be used to filter, and to sort the results, and finally to format the output.

We extracted, e.g., information about subcategorization properties of adjectives with respect to two different constructions:

1. adjective + PP
2. adjective + verb + clause

The extracted data can be used not only to determine subcategorization frames, but also to determine selectional preferences and possibly (related) distributional variations resulting from scrambling.

We extracted adjective + PP combinations out of a secure context, i.e., a context, where it is clear that the PP belongs to the adjective. We used complex APs embedding PPs as a source. We sort out APs with deverbal adjectives as head, which are marked by the feature \textit{\text{vder}}. Deverbal adjectives are not real adjectives, but verb participles used as adjectives. Thus, information gathered for deverbal adjectives is information about the underlying verb rather than about the adjective.

In the second experiment we extracted adjectives in predicative(-like) constructions subcategorizing a clauses. The resulting data can be used to extract different kinds of information:

- subcategorization patterns
- collocations (adjective+verb)
- distributional and selectional preferences
- semantic classes of adjectives

The combination of the annotation provided by YAC with a variety of relatively simple extraction queries allow a fine-grained distinction which gives a good picture about the properties and subcategorization information of adjectives in predicative(-like) constructions. More detailed investigation of the results can also provide a basis for semantic clustering.

The annotation of YAC also provides a good basis for the extraction of different types of named entities. In the first level of YAC a variety of special noun chunk rules is applied which are used to annotate named entities. Lexical-semantic properties annotated as feature attributes for the chunks can be used
to distinguish between the different types of named entities. As the structures are already annotated off-line, the extraction queries do not have to perform any kind of online text analysis but simply have to search for structures with a specific feature attribute.
Bibliography


Bibliography


NEGRA Project. NEGRA corpus home page. [http://www.coli.uni-sb.de/sfb378/negra-corpus/negra-corpus.html](http://www.coli.uni-sb.de/sfb378/negra-corpus/negra-corpus.html).


Bibliography


TIGER. The TIGER Project. http://www.ims.uni-stuttgart.de/projekte/TIGER/.


cischen Annotation großer Text-Korpora des Deutschen. In Proceedings of
the Work-Shop ‘Werkzeuge zur automatischen Analyse und Verarbeitung von

for nlp lexica. In Proceedings of the First International Conference on Language
Resources and Evaluation, Volume 1, Granada, pp. 129–134.


German. In Proceedings of the Sixth Conference on Natural Language Learning
(CoNLL 2002), pp. 56–62.

uni-tuebingen.de:8080/release/cass.html.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptoti
cally optimum decoding algorithm. IEEE Trans. on IT 13(2).

restricted English text. Technical Report 24, Department of General Linguis
tics, University of Helsinki, Helsinki, Finland.


a surface-syntactic parser of English. In U. Fries, G. Totti, and
P. Schneider (Eds.), Creating and using English language corpora. Amsterdam
and Atlanta: Rodopi.

al representation for parsing purposes. In Proceedings of the Seventh Confer
cence of the European Chapter of the Association for Computational Linguistics,

parser. In Proceedings of the Sixth Conference of the European Chapter of the
Association for Computational Linguistics (EACL'93), Utrecht, pp. 394–403.
Association for Computational Linguistics.

Zinsmeister, H. and U. Heid (2003). Identifying predicatively used adverbs by
means of a statistical grammar model. In Proceedings of Corpus Linguistics
2003, Lancaster, UK.

Parses for Treebank Annotations. In Proceedings of the LFG02 Conference,
CSLI Publications.
Bibliography