Morpho-Syntactic Annotation and Dependency Parsing of German

von

Julia Trushkina

Philosophische Dissertation
angenommen von der Neuphilologischen Fakultät
der Universität Tübingen

am 13. Dezember 2004

Tübingen

2004

Gedruckt mit Genehmigung der Neuphilologischen Fakultät
der Universität Tübingen

Hauptberichterstatter:	Prof. Dr. Erhard W. Hinrichs
Mitberichterstatter:	Prof. Dr. Uwe Mönnich
Dekan:	Prof. Dr. Joachim Knape

to my parents

Abstract

The parsing of natural language relies on the syntactic characteristics of words. The part of speech category is one of the most common sources of information in parsing. In the parsing of highly inflectional languages, morphological information, such as *case, number* and *gender*, also plays an important role. It helps to resolve syntactic ambiguity in shallow parsing and is particularly useful in dependency parsing of languages with free word order, since it partly determines the argument structure of the sentence.

For German, a highly inflectional language with partially free word order, the problem of assigning morpho-syntactic categories, such as part of speech, *case, number, gender, person, tense* and *mood*, i.e. the problem of morpho-syntactic annotation, is complicated by the high ambiguity inherent in tokens. Moreover, the partially paradigm-dependent case syncretism of this language makes the problem particularly intricate.

This thesis is concerned with the automatic morpho-syntactic annotation of German. Different approaches to the task are investigated in this thesis. A hybrid system with rule-based and statistical modules that combines the relative strengths of the rule-based and statistical methods involved is presented. The rule-based module is based on the Xerox Incremental Deep Parsing System and provides a novel constraint-based framework that integrates phrase-internal concord rules and phrase-external syntactic heuristics into one uniform architecture. The rule-based module successfully reduces the candidate analyses provided by a morphological analyzer. The statistical module is based on a novel use of probabilistic phrase-structure grammars for morpho-syntactic annotation. The module resolves the remaining cases of ambiguity, providing unambiguous and highly accurate output.

The usefulness of morpho-syntactic information is evaluated empirically in the creation of a dependency parser for German. The input to the parser is limited to tokens and their morpho-syntactic characteristics. The parser reaches state-of-the-art performance.

Acknowledgments

First of all, I would like to thank my advisor, Erhard Hinrichs, without whom this thesis would not be possible. Thank you for giving me the opportunity to come to Tübingen, which changed my life; for introducing me to computational linguistics; for teaching me many things; and for the endless hours of joint work and research which built the basis of this thesis.

I am also grateful to Uwe Mönnich, Fritz Hamm and Marga Reis, who kindly agreed to be in my thesis committee. Thank you for investing your time in reading the thesis and for your valuable and interesting feedback and suggestions on the final version of the thesis.

There is another colleague and friend of mine who has been an enormous help for me during all my years in Tübingen: Sandra Kübler. My admiration for your energy, your experience and your kindness, and many thanks for your patience in answering my countless questions on all possible subjects.

I would also like to thank my other current and former colleagues in Tübingen, as well as the students, for helping me in many ways and for being such a great team. I am particularly grateful to Tylman Ule for the inspiration in the experiments with probabilistic phrase structure grammars; to Heike Telljohann, Eva Klett, Silke Dutz and Emilia Ellsiepen for the morphological annotation of the treebank, which made the statistical experiments of the thesis research possible; to Gianina Iordachioaia for the friendship and the everyday support which made writing the thesis so much easier; to Beata Kouchnir, Sofia Katrenko, Martina Liepert, Frank Müller and Jorn Veenstra for the interesting and productive discussions; to Jochen Saile and Holger Wunsch for the computer and technical assistance; to Frank Richter, Lothar Lemnitzer, Beata Trawinski and Dale Gerdemann for encouragement and advice; to Armin Schmidt and Nathan Vaillette for the native speaker judgements and for the help with translations; to Jack Lauritsen for proofreading the thesis; to Cornelia Stoll for making administrative matters run smoothly; to Alexander Grebenkov, Ekaterina Ovchinnikova and Ekaterina Filippova for our philosophical debates and exchanges of knowledge; and to the whole SfS team for providing a stimulating environment and for sharing many exciting experiences.

I also want to express my appreciation to several people outside Tübingen who influenced my work on the dissertation in different ways: to Thorsten Brants and Dan Tufiş for the inspiring ideas; to Irina Azarova for giving me the first impulse towards a scientific career and for being always ready to help; to Kiril Simov for the support and advice; and to Karel Oliva and Vladimír Petkevič for the fruitful conversations.

I would also like to thank Xerox Research Centre Europe for the outstanding XIP system and particularly Claude Roux for being a prompt and valuable help in extending the system according to the needs of the thesis project.

I also thank all my friends for helping me through tedious hours.

Finally, and most importantly, I thank my best friend Zulfia Rakhmatullina and my family for their belief in me, for giving me freedom to be who I am, for supporting me in every situation and yet not letting me drift in the wrong direction, and for bearing with my moving thousands kilometers away from them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The focus of this thesis is the morpho-syntactic annotation of German and its application in the dependency parsing of German.

Morpho-syntactic annotation is the process of assigning each word in a text a tag from a previously defined set of morpho-syntactic labels, such as part of speech (POS) together with the appropriate morphological characteristics: *case, number, gender, person, tense* and/or *mood*. This process is often referred to as morpho-syntactic tagging.

The problem of the morpho-syntactic annotation of German was largely disregarded in the literature. With a few exceptions (Wothke et al. (1993), Steiner (1995), Lezius et al. (1998)), work in this area concentrated on pure POS-tagging. However, for other languages, particularly for languages with rich inflectional morphology, a substantial amount of research on morpho-syntactic annotation has been produced (see, among others, Oflazer and Tür (1996), Erjavec et al. (1999), Dienes and Oravecz (2000), Tufiş (2000), Hajič et al. (2001)). The aim of this thesis is to explore different approaches to the task of morpho-syntactic annotation of German and to provide a state-of-the-art morpho-syntactic tagger for German.

The current chapter introduces the problem of morpho-syntactic annotation and motivates its importance in the context of natural language applications. The difficulties associated with the problem are demonstrated in the chapter and the standard approaches and their drawbacks are briefly discussed. The chapter outlines the main ideas of the thesis. The overview of the thesis is given in the end of the chapter.

1.1 Statement of the problem

1.1.1 Motivation for morpho-syntactic annotation

Many natural language applications benefit from a deep analysis of the text. Thus, in current state-of-the-art machine translation and question answering systems, parsing has become a common module contributing to the successful realization of the task. Parsing has also found successful applications in Information Extraction, Speech Recognition and Text Summarization, among other areas.

For languages with a fixed word order, such as English, the POS characteristics of tokens together with their linear organization provide an adequate basis for parsing. Consider a sentence in (1):

(1)	The	father	told	his	children	a	story.
	DET	NOUN	VERB	POSDET	NOUN	DET	NOUN

In English, the first noun phrase (NP) corresponds to the `subject` of the verb, whereas the second and the third NPs have functions of `indirect` and `direct objects`, correspondingly.

For languages with a (partially) free word order, however, POS information proves insufficient. The English sentence in (1) can (with different focus variations) be translated into German into the following forms:

(2) a. Der Vater erzählt seinen Kindern eine Geschichte.
b. Eine Geschichte erzählt der Vater seinen Kindern.
c. Seinen Kindern erzählt der Vater eine Geschichte.

Unlike English, where only the `subject` can take the initial position in a sentence, in German any argument or adjunct of the verb, including `subject, direct` or `indirect object`, is grammatical in this position. In the field following the finite verb, the order of NPs is fixed but differs depending on whether the NPs are nominal or pronominal. Thus, the identification of grammatical functions in German sentences given only the word order and POS information is problematic.

The free word order of German is (partly) compensated by the inflectional morphology information of the tokens. This information helps to resolve the ambiguity created by possible permutations of NPs in the sentence: grammatical functions such as `subject`, `direct object`, and `indirect object` are strongly correlated with nominative, accusative and dative case, respectively. Therefore, the inclusion of case information into the set of word characteristics is crucial for the correct parsing of a sentence. Example 3 demonstrates the analysis of the German sentence given the POS and case values.

(3)	Der	Vater	erzählt	seinen	Kindern	eine	Geschichte.
	DET	NOUN	VERB	PRON	NOUN	DET	NOUN
	nom	nom		dat	dat	acc	acc
	NP-nom		VP	NP-dat		NP-acc	
	`subject`			`ind. object`		`direct object`	

Traditionally, the set of morpho-syntactic characteristics also includes such features as *number* and *gender* for nouns, adjectives, determiners, pronouns and verbs, *person* for pronouns and verbs, and *tense* and *mood* for verbs. This information finds applications in many areas including query answering, machine translation, speech synthesis, speech recognition, text summarization and the searching of large text data bases.

Apart from application-driven grounds for expanding the tagset with these features, there is an important practical reason: due to the interdependence between morpho-syntactic features, incorporation of them into a tagset helps annotation tools, taggers, to identify the case and POS values. Consider an example of German NP "*der Vater*".

The article "*der*" can be (a) *feminine, singular, dative* or *genitive*, (b) *any gender, plural, genitive* or (c) *masculine, singular, nominative.* Since German determiners and nouns have to agree in *case, number* and *gender*, number and gender information on the following noun, "*Vater*", which is *masculine, singular*[1], helps to resolve the case ambiguity.

1.1.2 Problematic issues in morpho-syntactic annotation

The morpho-syntactic annotation of German is complicated by high ambiguity inherent in German tokens. Consider the ambiguity of the above example sentence in (4):[2]

[1]The word "*Vater*" can have any case value, except for *genitive*

[2]The analyses for tokens are provided by the XRCE morphological analyzer for German.

(4)

Der	Det+Def+Fem+Sg+DatGen
Der	Det+Def+FMN+Pl+Gen
Der	Pron+Rel+Fem+Sg+Dat
Der	Pron+Dem+Fem+Sg+Dat
Der	Det+Def+Masc+Sg+Nom
Der	Pron+Dem+Masc+Sg+Nom
Der	Pron+Rel+Masc+Sg+Nom
Vater	Noun+Masc+Sg+NomAccDat
erzählt	Verb+Indc+2P+Pl+Pres+VVFIN
erzählt	Verb+Indc+3P+Sg+Pres+VVFIN
erzählt	Verb+PPast+VVPP
erzählt	Verb+Imp+2P+Pl+VVFIN
erzählt	Adj+PPast+Pred+ADJD
seinen	Pron+Poss+Pl+FMN+Dat+POSPRO
seinen	Pron+Poss+Sg+Masc+Acc+POSPRO
seinen	Det+Poss+Pl+FMN+Dat+St+POSDET
seinen	Det+Poss+Sg+Masc+Acc+St+POSDET
seinen	NAdj+Poss+Pl+FMN+Nom+Wk+NOUN
seinen	NAdj+Poss+Pl+FMN+Gen+Wk+NOUN
seinen	NAdj+Poss+Pl+FMN+Acc+Wk+NOUN
seinen	NAdj+Poss+Pl+FMN+Dat+Wk+NOUN
seinen	NAdj+Poss+Sg+Masc+Gen+Wk+NOUN
seinen	NAdj+Poss+Sg+Masc+Acc+Wk+NOUN
seinen	NAdj+Poss+Sg+Masc+Dat+Wk+NOUN
seinen	NAdj+Poss+Sg+Neut+Gen+Wk+NOUN
seinen	NAdj+Poss+Sg+Neut+Dat+Wk+NOUN
seinen	NAdj+Poss+Sg+Fem+Gen+Wk+NOUN
seinen	NAdj+Poss+Sg+Fem+Dat+Wk+NOUN
Kindern	Noun+Neut+Pl+Neut+Dat
eine	Det+Indef+Fem+Sg+NomAcc
eine	Pron+Indef+Fem+Sg+NomAcc
eine	Adj+Indef+FMN+Sg+Nom
eine	Adj+Indef+FN+Sg+NomAcc
Geschichte	Noun+Common+Sg+Fem+NGDA+NOUN
.	Punct+Sent+SENT

The tokens are not only ambiguous in respect to their POS category (determiner – `Det`, pronoun – `Pron`, adjective – `Adj`), but also as to their morphological characteristics. Beside separate morphological tags, the analyses include joint morphological tags, such as `NomAccDatGen` standing for any case and `FMN` standing for any gender. Each joint tag, thus, expands into several distinct analyses.

Case syncretism, i.e. the phenomenon that one and the same form may express two or more cases, makes the problem even harder. Compared to other languages where case syncretism tends to conflate the same cases across different nominal paradigms, German case syncretism is particularly intricate since it is often paradigm-dependent. Table 1.1 represents an example of different nominal paradigms in German.

Nominative	Genitive	Dative	Accusative
Blume	∼	∼	∼
Bär	∼en	∼en	∼en
Bilder	∼	∼n	∼
Vater	∼s	∼	∼
Name	∼ns	∼n	∼n

Table 1.1: An example of German nominal case syncretism

Table 1.2 presents ambiguity rates for German estimated on the data in the thesis experiments. The average number of analyses is counted as the ratio between the number of analyses assigned to the tokens in the text and the total number of tokens in the text. The percentage of ambiguous tokens in the data is provided in column 3.

For comparison, ambiguity rates reported in the literature for other languages, as well as for the pure POS tagging of German, are included in the figure.[3]

What this comparison shows is that the morpho-syntactic annotation of German constitutes a much harder task than the same problem for other languages or for the pure POS tagging of German. The tagset for annotation is the largest tagset used in the experiments with morpho-syntactic annotation. The average number of analyses is almost double that of Czech and is at least by a factor of 3 higher than for the other languages. This

[3]The Czech data have been described by Hajič and Hladka (1997), Turkish – by Oflazer and Tür (1996), English – by Tapanainen and Voutilainen (1994), Romanian – by Tufiş (2000), and Hungarian – by Tufiş et al. (2000).

language	average # analyses	ambiguous tokens	tagset size
German	5.80	66.61%	1317
Czech	3.65	not available	1171
	2.36	not available	882
Turkish	1.83	50.66%	not available
English	1.77	not available	139
German (POS)	1.77	39.57%	54
Romanian	1.71	38.17%	410
Hungarian	1.33	31.90%	> 1265

Table 1.2: Ambiguity of German data in comparison to other languages

is reflected in the percentage of ambiguous tokens, which is almost twice as big compared to Romanian, English, Turkish and Hungarian.

1.1.3 Previous approaches to morpho-syntactic annotation

The tagging problem, i.e. the problem of identifying the (morpho-)syntactic class of a word, was first explored for English by Harris (1962) and Klein and Simmons (1963). They employed rule-based methods that rely on hand-crafted constraints about sentence grammaticality. Such methods imply disambiguation of tokens on the basis of the surrounding context: a reading can be eliminated if it creates an ungrammatical construction, as in (5):

(5) He opens a can.

A verbal reading of word "*can*" can be eliminated, since a sequence of an article and a following verb is ungrammatical in English.

Rule-based methods are still commonly used: rule-based systems were developed, among others, for English by Voutilainen (1995*b*), French by Chanod and Tapanainen (1995), Turkish by Oflazer and Kuruöz (1994), and Swahili by Hurskainen (1996). Rule-based methods usually provide high annotation accuracy and enable the reflection of linguistic insights, such as subject-verb agreement, in the rules. A widely admitted drawback of such systems is that their creation is time- and labor-consuming. An automatic rule-based tagger was developed by Brill (1992). The tagger automatically acquires its rules and tags from training data.

An alternative to rule-based methods are probabilistic methods, on which the majority of currently used taggers are based (to mention only a few, commonly used taggers were developed and described by Church (1988), Cutting et al. (1992), Brants (1998)). Such taggers are trained on ambiguously or unambiguously tagged texts to learn the lexical and contextual probabilities of tokens, i.e. the probability of observing a word given a tag and the probability of observing a tag given a context (usually one or more preceding tags), correspondingly. The tag sequence that maximizes the product of lexical and contextual probabilities provides analyses for tokens in the sentence.

Another probabilistic framework applied to tagging is the Maximum Entropy framework. In this framework, information about a token and its context is incorporated in the form of features. An argument that maximizes the product of weighted features corresponds to a tag for the token.

Tagging systems based on symbolic machine learning techniques have also become one of the dominant paradigms for the task (Daelemans, Zavrel, Berck and Gillis (1996), Magerman (1995), Benello et al. (1989), Schmid (1994*b*)). They include memory-based learning, decision trees and neural network based methods.

With a few notable exceptions (Dienes and Oravecz (2000), Hajič and Hladka (1997), Tufiş (2000)), most tagging approaches have focused on tagging with fairly small tagsets, distinguishing only between parts of speech, but not taking into account more fine-grained distinctions of morphology. Designing tagging systems for languages with rich inflectional morphology, such as German, Czech, Hungarian and Romanian, encounters difficulties that are comparatively harmless for pure POS tagging but constitute a major obstacle for morpho-syntactic annotation. These difficulties concern the size of a tagset required for such languages and the non-local nature of the contextual cues which have to be taken into account for morphological disambiguation.

Thus, the best morpho-syntactic tagging system for German developed by Lezius et al. (1998) is based on the trigram algorithm described by Church (1988). While the system achieves state-of-the-art performance for the task of POS-tagging of German (95.9%), the results for the task of morpho-syntactic tagging are much lower: 84.7%.

The limitation of n-gram models that creates such notable difference in model performance for the two tasks lies in the following: n-gram taggers consider only sequences of **n** words and their candidate tags, i.e. very local contexts, as the basis for determining the most likely sequence of tags for the sentence. This Markovian assumption proves harmful for decisions that crucially require larger context windows. *Case, person* and *number* information is precisely of this nature, since successful disambiguation needs to rely on genuinely syntactic phenomena such as subject-verb agreement, valency of main verbs, and morphological features of other nominal elements in the sentence.

Machine learning methods, which hold on the same principle of using a small context window, encounter the same problems. Increasing the window size unavoidably leads to the sparse-data problem: a sequence of `n+1` tokens occurs less often than a sequence of `n` tokens, and thus, more data is needed for reliable estimation of the probabilities. For languages with limited data resources, methods requiring vast amounts of annotated data are inapplicable. Due to the difficulty of the task of automatic morpho-syntactic annotation, the necessary resources have not been yet created for German. A vicious circle arises: existing automatic methods require vast amounts of data for successful realization of the task, while the data are not available. Quick and accurate annotation of data is, on the other hand, impossible without automatic taggers. New methods which would solve the problem are needed.

1.1.4 Main ideas of this thesis

In this thesis, alternative methods to the morpho-syntactic annotation of German are explored. The power of the rule-based approach is re-assessed and the ability of the rule-based approach to convey fine-grained morpho-syntactic disambiguation with high precision is demonstrated. A novel constraint-based framework that integrates phrase-internal concord rules and phrase-external syntactic heuristics into one uniform architecture is described.

Different statistical approaches to morpho-syntactic tagging are further explored and the main obstacle to successful application of the n-gram based methods to the task is demonstrated. The thesis presents a novel statistical approach to tagging, which utilizes probabilistic phrase-structure grammars (PCFGs) for the problem at hand. Due to their ability to incorporate global structural information, PCFGs provide a suitable alternative to n-gram models and yield acceptable results even for moderate amounts of training data. It is also shown that tree transformations of the training data constitute a crucial step in optimizing the performance of the PCFG model for the task of morpho-syntactic annotation.

Finally, a hybrid tagging system for German is presented. The system combines the powers of rule-based and statistical methods.

1.2 Overview of the thesis

Following this introduction, Chapters 2 and 3 present the theoretical background of research reported in the thesis. Chapter 2 discusses different approaches to morpho-syntactic annotation and provides a comprehensive literature review for the current task. Moreover, the chapter demonstrates problematic issues in the application of the existing methods to German. Chapter 3 provides a necessary background for one of the possible areas of application for the use of morpho-syntactic information: dependency parsing. The main ideas and principles of dependency syntax, as well as existing German models are discussed in this section.

Chapter 4 presents the Xerox Incremental Parsing System – a formal framework which has served as a basis for creation of the rule-based model of morpho-syntactic annotation, as well as an underlying formalism of the implementation of a German dependency parser. The general architecture of the system, its functionality and different types of rules present in the system are described.

Chapter 5 introduces the TüBa-D/Z treebank: a data source for the experiments and the evaluation of the designed morpho-syntactic annotation system and the dependency parser of German reported in the thesis. Apart from providing the data for the research, the treebank partly determined decisions in the development of the dependency parser, since the underlying principles of the treebank annotation scheme guided the annotation produced by the parser. Concepts, principles and structures of the treebank are discussed in this chapter.

The core chapter of the thesis, Chapter 6, describes different models for morpho-syntactic annotation of German. This chapter is based on previously published work co-authored with Erhard Hinrichs. Section 6.1 presents a rule-based model based on constraint satisfaction techniques: sequential disambiguation of tokens by sophisticated hand-written rules that leads to high accuracy of annotation. In Section 6.2, statistical methods to morpho-syntactic annotation are discussed. It is shown that standard techniques of n-gram taggers such as TnT are inadequate for this task, but that probabilistic context-free grammars provide a suitable alternative that yields acceptable results even for moderate amounts of training data. Section 6.3 compares the rule-based and statistical methods and discusses their advantages and drawbacks. Section 6.4 presents a hybrid model of rule-based and statistical modules that combines the relative strengths of the methods involved. Section 6.5 concludes the chapter.

In Chapter 7, the use of morpho-syntactic information is demonstrated in its application in dependency parsing. The chapter considers the German dependency parser developed on the basis of Xerox Incremental Parsing System. A thorough description of the parser and its different modules together with the detailed evaluation of parser performance is provided.

In Chapter 8, the German Incremental Parsing System (GRIP) is presented. It consists of the morpho-syntactic tagger described in Chapter 6 and the dependency parser described in Chapter 7. The architecture of the system is presented and the evaluation of the system is provided.

Chapter 9, finally, summarizes the research reported in the thesis and sketches directions for future work.

Chapter 2

State of the art in morpho-syntactic annotation

The broad application of tagging in natural language processing (NLP) gave rise to widespread interest in the task. Various methods have been explored in the literature and numerous successful tagging systems have been reported. This chapter aims at supplying a survey of the state of the art for the task of morpho-syntactic tagging. Section 2.1 provides a general overview of different approaches to POS and morpho-syntactic tagging. In sections 2.2 and 2.3, the main ideas of existing methods are presented more thoroughly and each method is exemplified with a detailed description of one or several taggers based on the methodology discussed. Section 2.4 examines the advantages and drawbacks of the approaches and discusses the applicability of the methods to the task of morpho-syntactic annotation. Section 2.5 follows this discussion by a description of hybrid methods which aim at combining the strengths of different techniques and presents extensions of methods relevant for the task of morpho-syntactic annotation. Section 2.6 summarizes the state of the art in tagging of German and section 2.7 concludes the chapter.

2.1 General overview of the approaches

Two main methodologies in tagging can be distinguished. They differ with respect to the kinds of information they are based on.[1] Approaches of the first methodology rely on linguistic knowledge which is either recorded by a

[1] The distinction is based on the dichotomy presented in Voutilainen (1995*b*).

linguist or automatically extracted from corpora. This knowledge is incorporated in the form of constraint rules which capture regularities between possible analyses of a token and its surrounding context. Serial application of rules allows for the resolution of the ambiguity. A basic example of techniques of this methodology is the disambiguation of a token "*play*" in the phrase "*the play*". The token has a nominal and a verbal readings. The verbal reading, however, can be eliminated, since a sequence of a determiner and a following verb is ungrammatical for English. Approaches of this kind are called *rule-based* or *constraint-based* approaches. Depending on the source of the rules, hand-crafted and automatic rule-based approaches are distinguished. In the former, rules are designed by a linguist. In the latter, a tagger acquires rules automatically in an iterative way by comparing its own output to a correctly annotated text.

Approaches of the second methodology rely on frequency-based information automatically derived from corpora. Two essential sources of information are used (Manning and Schütze (1999)): (a) tags and/or lexemes in the surrounding context (contextual, or syntagmatic, information) and (b) set of possible tags for the token being tagged (lexical information). Contextual information allows the selection of a tag which is more likely in a given context. Thus, in the context "*they play jazz*", the verbal reading is preferred over the nominal reading for "*play*", since a pattern `Pron Verb Noun` is more common than a pattern `Pron Noun Noun`. Lexical information accounts for the fact that possible analyses of a token are usually not equally distributed. For example, "*jazz*" has a verbal reading along with a nominal reading, but the nominal reading is much more frequent. All existing approaches to automatic data-driven tagging in some way make use of a combination of these two types of information. They collect statistics about lexical and contextual information from the corpora and use them to assign tags to tokens in a sentence.

Frequency-based data-driven approaches differ in the way frequency information is used. Probabilistic (also called *statistical* or *stochastic*) methods are based on calculation of statistics of the events and maximizing the probabilities in creation of a model and in tagging. Markov model taggers maximize the product of lexical and contextual probabilities, where context is restricted to preceding tokens (usually 2, maximally 3). Maximal entropy taggers incorporate lexical and contextual information in a set of binary *features*, where each feature specifies a condition on the characteristics of the focus word/tag and the surrounding words/tags. Probability of a tag is calculated as a product of weighted feature values.

Symbolic machine learning approaches, on the other hand, do not explic-

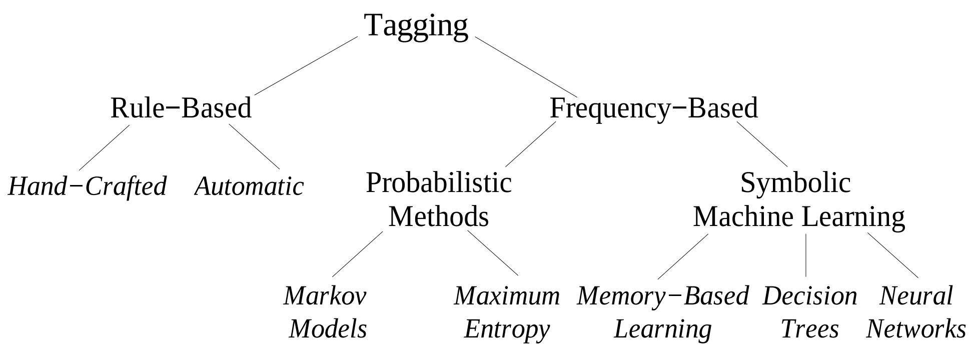


Figure 2.1: Approaches to tagging

itly use probabilities in the hypothesis and differ in the way of representation of the information collected from corpora (instance memory-bases, decision trees or neural networks) and in the procedure of the assigning tags to new data.

The diagram in Figure 2.1 represents relations between the approaches to tagging mentioned above. In reality, the diversity of tagging approaches is more complex, since many tagging systems unify aspects of different approaches. A typical example of such a system is the Transformation-Based Tagger (Brill (1995*a*)), which can be assigned to both rule-based approaches, as its tagging procedure is based on rule application, and to symbolic machine learning approaches, since it acquires its transformations by learning from corpora. Moreover, for many of the types in the figure, further distinction into supervised and unsupervised tagging is possible. This distinction concerns the amount of prior knowledge provided to the tagger. Supervised approaches rely on pre-tagged corpora. The task is then to create a model that most accurately predicts tags in the training corpus. Unsupervised approaches, on the other hand, do not require a pre-tagged corpus. They make use of unannotated data and a lexicon which contains possible tags for each token to induce the regularities needed for tagging: context rules for automatic rule-based approach and probabilistic information for frequency-based approaches.

The remainder of this chapter provides a more detailed discussion of the approaches, as well as their comparison.

2.2 Rule-based approaches

Pioneering work in tagging done in the 1960's employed the hand-crafted rule-based method (Harris (1962), Klein and Simmons (1963)). A tagger TAGGIT described by Greene and Rubin (1971) and based on the same method was used to automatically pre-tag the Brown corpus, the first of the modern computer readable corpora. However, performance of rule-based taggers was rather low: for example, TAGGIT provided correct analyses only for 77% of tokens. This non-optimal performance lead to a decreased interest in the rule-based approach.

Interest in rule-based approaches reemerged in the 1990's with the development of the Constraint Grammar formalism for English (Karlsson (1990), Karlsson et al. (1995)). The formalism demonstrated high performance and stimulated the development of rule-based taggers for other languages (Oflazer and Kuruöz (1994), Chanod and Tapanainen (1995), Hurskainen (1996)). Later on, the EngCG (English Constraint Grammar) morphological disambiguator based on the Constraint Grammar framework was created (Voutilainen (1995*b*)). The EngCG is currently one of the best taggers for English.

The main drawback of the hand-crafted rule-based taggers lies in a great amount of labor and time required for the development of such taggers. The rule-based tagger described by Brill (1992) overcomes this limitation: the rules are automatically acquired from corpora.

Below, the two types of the rule-based method are described on the example of the EngCG tagger and the Transformation-Based Learning Tagger, also known as Brill Tagger (Brill (1995*a*)), a later version of the tagger presented by Brill (1992).

2.2.1 The EngCG Tagger: A Hand-Crafted Rule-Based Tagger

The EngCG Tagger was developed in 1989-1993 at Helsinki University primarily as a tagger for standard written English. It consists of the following sequential modules:

1. a tokeniser
2. a morphological analyser
3. a rule-based disambiguator

The tokeniser represents a rule-based system that identifies words, punctuation marks, document markers and multiword expressions – idioms and modifier-head expressions. Thus, the tokeniser divides a string into processing units – tokens, which are then further processed by the morphological analyser.

The morphological analyzer contains a two-level lexicon of approximately 90,000 entries and a rule-based heuristic analyzer of unknown words (guesser). The analyses for tokens are first looked up in the lexicon. If no entry is available for the token in the lexicon, the guesser tries to identify the morpho-syntactic characteristics of the token based on the heuristics about derivational morphological features of the token. For example, if a token ends with the suffix "*-ously*", the guesser provides an adverbial analysis for this token. If the analyses cannot be identified by heuristics, a nominal reading is given to the token. Experiments of Tapanainen and Voutilainen (1994) have shown that 5% of all word-form tokens are not present in the lexicon, but the guesser provides a correct analysis for 99.5% of those tokens.

The tagset of the morphological analyzer includes 139 tags mainly for POS, case, number, tense and mood values, as well as some more subtle syntactic subcategorizations (e.g. relative vs. demonstrative pronouns). Analyses for tokens are represented as tag combinations. Altogether, the morphological analyzer produces about 180 different tag combinations. The average number of analyses after the application of the morphological analyzer ranges between 1.7–2.2 analyses per token.

In the second version of the EngCG (Voutilainen (1997)), an additional post-analyzer module has been built. This module introduces missing appropriate readings to the tokens by context-sensitive replacement mechanism. For example, such module replaces all readings containing the infinitival tag `INF` with the tag sequence `<CMH> N NOM SG` (standing for *common noun, nominative, singular*) if all four conditions on the context are satisfied:

1. the first word to the left is an unambiguous determiner, or genitive, or preposition, or title-word (such as *Mr, Mrs* or *Dr.*);

2. the first word to the left does not contain the tag `Rel` (for relative pronouns and relative determiners) or the tag `INDEP` (for certain genitives acting as a noun phrase head, e.g. "*theirs*");

3. the word itself is not a form of "*let*", nor does it contain tags from the set `OPEN-NOMINAL` (e.g. nouns), or the set `AUX-MOD` (e.g. auxiliary verbs), nor is it a preposition or a conjunction;

4. the first word to the right does not contain readings of an article or a genitive pronoun, nor does it have a value *accusative* for case.

Such a heuristic corrects, for example, the set of analyses for the token "*wrestle*" in the context "*a wrestle with gravitation*".[2]

After introducing the ambiguity on the tokens, the last module of the system is activated to disambiguate tokens in the text. The disambiguation module is a system of pattern-action rules which specify a context of rule application and an action produced if the context constraints are specified. The rules are able to delete or select a specific reading on a token. An example rule presented in Voutilainen (1997) is shown in (6):

```
(6)  SELECT (INF)
                (-1 DO)
                (NOT -1 PTCPL)
                (NOT 0 PROPER OR DO-OBJ-NOUN) ;
```

This example rule selects an infinitival reading if the token is preceded by a non-participial form of "*do*". Additional constraints are specified on the token itself: it should not be a proper noun, nor a noun that typically occurs as object of "*do*", e.g. "*credit*".

The rule in (6) exemplifies a lexicalized rule, i.e. a rule in which at least one of the constraints directly refers to a word-form of a token. However, Voutilainen (1995*b*) claims that the majority of rules in the system are based on a small number of essentially syntactic generalizations about the form of the phrases (nominal, prepositional etc.) in a sentence. 71% of constraints are reported to be global: i.e. they operate on a context that extends beyond the neighboring words.

The rule-based disambiguator consists of 5 sequentially applied subgrammars. First 3 subgrammars contain highly reliable "grammar-rules" which avoid risky predictions. Application of these subgrammars results in high precision (99.7%) but leaves 3-7% of all tokens ambiguous, which corresponds to 1.04-1.08 alternative analyses per output token. The application of other 2 subgrammars is optional and leads to further disambiguation of tokens at the expense of decrease in precision. The constraints of these 2 subgrammars represent powerful syntactic heuristics which resolve approximately 50% of remaining ambiguity, increasing the overall error rate to about 0.5%.

[2]The example taken from Voutilainen (1997) is provided here in a shortened form.

2.2.2 The Brill Tagger: An Automatic Rule-Based Tagger

The Brill tagger is based on a transformation-based error-driven learning method. The general idea of the method consists of the following: at first, a text is tagged with a simple initial-state annotator; during the next stage, the output of the annotator is compared to a standard provided by a manually annotated corpus; based on the comparison, transformational rules are learned which correct the errors of the initial-state annotator. New text can be tagged by sequential application of the initial-state annotator and of the acquired transformations. The pre-requisite for transformation-based tagging consists of a pre-tagged training corpus and templates of admissible transformations. For the unsupervised version of the tagger (Brill (1995*b*)), an unannotated corpus and a dictionary are required, as well as templates of admissible transformations.

The complexity of the initial-state annotator can range from random assignment of tags to sophisticated procedures based on probability distributions of tags and morphological characteristics of tokens. The initial-state annotator described in Brill (1995*a*) assigns each word its most likely tag as indicated in the training corpus.

The transformation templates that the tagger uses to learn real transformational rules are of the form:

1. If a word is tagged **a** and it is in context **C**, then change that tag to **b**, or
2. If a word is tagged **a** and it has lexical property **P**, then change that tag to **b**, or
3. If a word is tagged **a** and a word in region **R** has lexical property **P**, then change that tag to **b**.

The context **C** and the region **R** are restricted to context window of length 3, i.e. maximally three words in the immediate neighborhood of the token can be taken into account.

Example rules learned by the tagger for English are listed in Table 2.1. The first rule re-tags common nouns (`NN`) as verbs (`VB`) in a position after a modal verb (`MD`). The second rule applies to non-capitalized common nouns, changing them to proper nouns. The last rule states that if a word is tagged as a past participle (`VBN`) and is preceded by a capitalized word, then it should be re-tagged as a past tense verb form (`VBD`).

For each rule created from a set of templates, its performance is evaluated as the improvement on the tagging accuracy after the rule application. This

Source tag	Target tag	Condition
`NN`	`VB`	previous tag is `MD`
`NP`	`NN`	current word is not capitalized
`VBN`	`VBD`	previous word is capitalized

Table 2.1: Example rules learned by a transformation-based tagger.

	Left to Right	Right to Left
Immediate	**a b a b**	**a b b b**
Delayed	**a b b b**	**a b b b**

Table 2.2: Tagger output depending on the application of transformations.

improvement is counted as a difference between the number of corrected errors and the number of new errors caused by the rule. The rule with the best performance is added to the final list of rules of the tagger and the procedure is repeated until there is no rule that reduces the error rate by more than a pre-specified threshold. Such a learning algorithm selects the list of rules and predefines the order of their application as the order in which rules were added to the list.

Two decisions have to be made about the application of the rules. First, in which direction the transformations are applied to a corpus: right to left or left to right. Second, whether a transformation is applied immediately or only after the whole corpus has been examined. Consider a sequence "**a a a a**" and a rule "*Change tag* **a** *to* **b** *if preceded by* **a**". As Table 2.2 show, the output of the tagger differs depending on the decisions made.

An unsupervised version of a transformation-based tagger is described in Brill (1995*b*). Both the initial-state annotator and the learning algorithm, as well as the set of rule templates, are different from those of the supervised model.

The initial-state annotator assigns each token all possible tags as specified for the token in the dictionary. The template rules indicate a set of source tags (as opposed to a single source tag in the supervised model). The template set is additionally restricted to templates with the context window of size 1. Unlike the learning algorithm of supervised tagging that needs gold tags from the pre-tagged corpus as a basis for comparison, the learning algorithm of unsupervised tagging relies on the tags currently assigned to the tokens by the tagger. The algorithm takes advantage of the fact that many words in the corpus are initially unambiguous. Instead of comparing a candidate tag to a gold tag from a pre-tagged corpus, the candidate tag is compared to tags of unambiguous words occurring in the same context. A rule is considered reliable if one of the source tags appears much more frequently as a tag of unambiguous words in the specified context than the others. This difference in frequency of source tags serves as a basis for the scoring function that evaluates the rule performance.

The tagger was originally designed for English and evaluated on the Brown corpus. The performance of the supervised and unsupervised taggers reaches 96.3% and 95.6%, correspondingly. Volk and Schneider (1998) applied the supervised version of the tagger to the task of pure POS-tagging of German. They report an achieved accuracy of 94.75%.

2.3 Frequency-based approaches

Frequency-based approaches to tagging rely on frequency information extracted from data. The popularity of this type of approach lies in the general applicability of the methods: given the data, a tagger can be easily retrained for a different domain or a different language. Depending on the way the frequency information is used by a tagger, probabilistic methods and symbolic machine learning methods are distinguished. In probabilistic methods, probabilities of events are calculated and the most likely output is chosen. In symbolic machine learning methods, classification of objects is pursued in a non-probabilistic fashion. The two types of approaches are discussed below.

2.3.1 Probabilistic Methods

Markov Modeling

Markov Models taggers dominate the field: they are easy to build, they are fast and accurate. Both supervised and unsupervised Markov model taggers are widely used. Below, a short introduction into Markov models is given, and their application to tagging is discussed. A Markov model tagger is exemplified by the Trigrams'n'Tags tagger (TnT), one of the best POS taggers currently available.

Markov Models

A Markov model is a probabilistic model based on Markov chains. A Markov chain is defined as a stochastic process with the *Markov Property*, i.e. as a sequence of random variables in which the probability distribution of possible values of the next variable depends only on the value of a current variable and is independent of the values of previous variables. The Markov Property is formally stated as

$$P(X_{t+1} = s_{i_{t+1}} \mid X_1 = s_{i_1}, \ldots, X_t = s_{i_t}) = P(X_{t+1} = s_{i_{t+1}} \mid X_t = s_{i_t})$$

where $X = (X_1, \ldots, X_T)$ is a sequence of variables which take discrete values in some finite set $S = \{s_1, \ldots, s_N\}$, a state space.

A Markov chain can be described by a stochastic transition matrix A:

$$a_{ij} = P(X_{t+1} = s_j \mid X_t = s_i)$$

where a_{ij} represent transition probabilities from state X_i to state X_j. The probabilities are non-negative and for each state X_i, the sum of the outgoing transition probabilities equals 1:

$a_{ij} \geq 0, \forall i, j; \quad \sum_{j=1}^{n} a_{ij} = 1, \forall i.$

Additionally, the probabilities of possible initial states are needed: $\pi_i = P(X_1 = s_i)$. However, if an extra initial state s_0 is added and if it is specified that the model always starts at this state, probabilities π_i can be incorporated into the transition matrix A.

In many applications, including natural language applications, Markov chains are restricted to *homogeneous* or *time invariant* Markov chains, in which for all *i, j*, transition probabilities

$$a_{ij} = P(X_{t+1} = s_j \mid X_t = s_i)$$

are independent of *t*.[3]

Figure 2.2 shows an example of a Markov chain. Here, the states are represented as circles with state labels inside the circles and transitions are depicted as arrows connecting the states, with transition probabilities on the arrows. The initial state of the model is indicated by an incoming *start* arrow. Zero probability transitions are omitted from the picture.

A Markov model is a generalization of a Markov chain where each state is associated with a probabilistic output function. More formally, a Markov model consists of:

1. a finite set of states $S = s_0, \ldots, s_n$ with a unique initial state s_0;

[3]For non-homogeneous Markov chains a_{ij} are not discrete values, but functions.

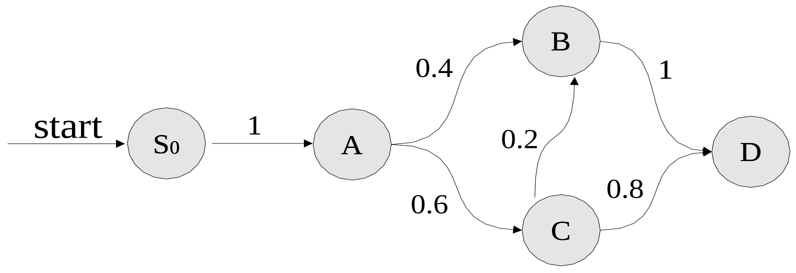


Figure 2.2: An example of a Markov chain.

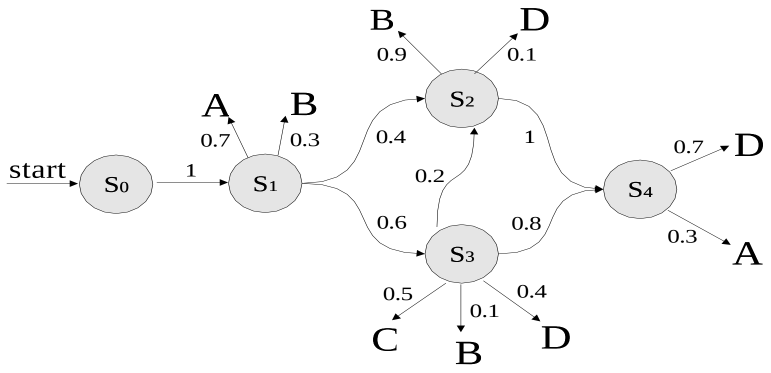


Figure 2.3: An example of a Markov model.

2. an $(n+1)\times(n+1)$ transition matrix $A = [a_{ij}]$, where $a_{ij} = P(X_{t+1} = s_j \mid X_t = s_i)$, $a_{ij} \geq 0, \forall i, j \in S$ and $\sum_{j=1}^{n} a_{ij} = 1, \forall i \in S$;

3. an output alphabet $K = k_1, \ldots, k_m$;

4. an $n \times m$ symbol emission matrix $B = [b_{ik}]$, where $b_{ik} = P(O_t = k \mid X_t = s_i)$, $b_{ij} \geq 0, \forall i \in S, k \in K$ and $\sum_{k=1}^{m} b_{ik} = 1, \forall i \in S$.

An example of a Markov model is presented in Figure 2.3.

Two types of Markov models are distinguished: *visible* (or *observable*) Markov models and *hidden* Markov models. The distinction is made based on whether states of the model are observable or hidden. An example of a visible Markov model is a model of sequences of tagged words: tuples *word-tag* correspond to states and the emitted symbols. A sequence of words with unknown tags can be modeled by a hidden Markov model: the output symbols, words, are observable, but the states, i.e. tags of the words, are hidden.

Given annotated training data, a (visible) Markov model learns the transition and emission probabilities by calculating relative frequencies of the events:

$$\hat{P}(k^i \mid k^j) = \frac{C(k^j, k^i)}{C(k^j)}$$

$$\hat{P}(s^i \mid k^j) = \frac{C(s^i, k^j)}{C(k^j)}$$

Here, k^i and k^j are elements in the output alphabet, s^i is a state, $C(k^j, k^i)$ is the number of occurrences of k^j was followed by k^i, $C(t^j)$ is the number of occurrences of k^j in the training data and $C(s^i, k^j)$ is the number of times k^j was emitted by s^i.

A problem arises if k^j did not occur in the training data. First, the quotients in the equations are undefined. This is solved by defining zero divided by zero to be zero. However, it still means that $P(k^i \mid k^j)$ and $P(s^i \mid k^j)$ equal zero for all k^i, which has an undesirable effect: a string of tags containing a substring $k^j\ k^i$ will have a zero probability. The problem can be solved by *smoothing* the probabilities. One way to do so is by adding further terms to the equation:

$$\hat{P}(k^i \mid k^j) = \lambda_1 \frac{C(k^j, k^i)}{C(k^j)} + \lambda_2 V$$

where V is a term added and λs are the weights given to each term.

If no annotated training data is available, there is no straightforward way of empirically deriving parameters of a Markov model.[4] However, parameters can be estimated with the *Baum-Welch* or *Forward-Backward algorithm* (Baum et al. (1970)). The algorithm starts with making an initial guess of the parameters and proceeds by estimating the use of each parameter. By re-calculating the probabilities of the parameters on the basis of the expected number of transitions among states and the expected number of symbol emissions for a given state, given the training data, a revised model is received which gives a higher probability of the given data. The re-estimation is repeated until a new model is no longer improving the approximation.

[4]In this case one deals with a hidden Markov model.

Once a model is trained, the most likely output sequence of the model can be calculated with the *Viterbi algorithm* (Viterbi (1967)). For each state in a model, the algorithm finds an incoming transition with the highest probability and keeps track of this probability and the state from which the transition comes (a "backtrace"). The mostly likely sequence is found by choosing a final state with the highest probability and back-tracking the most likely path to the initial state through the stored backtraces.

The amount of history encoded in the state space defines the order of a Markov model. In first-order Markov models, a state incorporates information about one category. Alternatively, a state can represent a list of categories of n previous states plus a category of the state itself. In such case, a model is called m^{th} *order* Markov model, where m is a number of previous states used to predict the next state. Markov models of order (n-*1*) are equivalent to n-gram models.

Markov Model Tagging

In Markov model tagging, a sequence of tags is viewed as a Markov chain, i.e. an assumption is made that a word's tag only depends on tags of the n preceding words and that this dependency does not change over time, i.e. the position in the sequence. Although the assumption does not strictly correspond to the reality[5], Markov model taggers have demonstrated a convincing success in the domain: the best Markov model taggers perform with an accuracy of 96-97% on the task of POS-tagging of English.

When a Markov model is applied to tagging, the states correspond to tags, a transition probability a_{ij} reflects a *contextual* probability $P(t_i \mid t_j)$ of a sequence of tags $t_j t_i$. States emit words with *lexical* probabilities $P(w \mid t)$. The task is to find the most likely sequence of tags $t_{1,k}$ for a string $w_{1,k}$, i.e. a tag sequence that maximizes the conditional probability $P(t_{1,k} \mid w_{1,k})$:

$$\underset{t_{1,k}}{argmax} P(t_{1,k} \mid w_{1,k}) = \underset{t_{1,k}}{argmax} \frac{P(t_{1,k})P(w_{1,k} \mid t_{1,k})}{P(w_{1,k})}$$

$P(w_{1,k})$ is constant for all $t_{1,k}$, so it is sufficient to find

$$\underset{t_{1,k}}{argmax} P(t_{1,k})P(w_{1,k} \mid t_{1,k})$$

Making the Markov assumptions, this formula can be rewritten as[6]

[5]Under such assumption, long-distance dependencies cannot be caught.

[6]Full derivation is presented in Appendix F

$$\underset{t_{1,k}}{argmax}\prod_{i=1}^{k} P(t_1 \mid t_{i-1})P(w_i \mid t_i)$$

The formula describes a first order Markov model, or a *bigram* model. It combines contextual and lexical probabilities. When a model is trained, the probabilities are known and the most likely tag sequence is found with the Viterbi algorithm.

For tagging purposes, Markov models are treated as hidden Markov models, since only words and not tags are known. For training purposes, however, the use of visible Markov models and HMMs differs. When tagged training data is available, both tags and words are known and a visible Markov model is used. Parameter estimation of a visible Markov model tagger (often called an *n-gram* tagger) is straightforward: the probabilities are at first derived from relative frequencies of sequences of tags and word-tag pairs and then smoothed. If only untagged data are available, HMMs are used to learn the probabilities with the Baum-Welch algorithm.

Markov models are one of the most widely used techniques in NLP and are one the most popular approaches in tagging, as the amount of Markov model-based taggers shows (Jelinek (1985), Church (1988), DeRose (1988), Kupiec (1989), Cutting et al. (1992), Kempe (1993), Merialdo (1994), Brants (2000)).

Trigrams'n'Tags tagger (TnT)

Trigrams'n'Tags (TnT) is a trigram tagger developed by Brants (2000). The states of the model represent tags, outputs represent words, transition probabilities depend on the states, i.e. pairs of tags. The main smoothing technique implemented by default is linear interpolation. Thus, a trigram probability is estimated as follows:

$$P(t_3 \mid t_1, t_2) = \lambda_1 \hat{P}(t_3) + \lambda_2 \hat{P}(t_3 \mid t_2) + \lambda_3 \hat{P}(t_3 \mid t_1, t_2)$$

where $\hat{P}$ are maximum likelihood estimates of the probabilities, and $\lambda_1 + \lambda_2 + \lambda_3 = 1$. The values of the λs are determined by deleted interpolation (Brown et al. (1992)).

Unknown words are handled by the method proposed by Samuelsson (1993) which is based on the suffix analysis. The information about capitalization of words is also added to the tagging scheme as an indicator function $c(w_i)$ with the value 1 if the word is capitalized and 0 otherwise. The information is incorporated into the contextual probability distributions. For example, instead of $P(t_3 \mid t_1, t_2)$, $P(t_3, c_3 \mid t_1, c_1, t_2, c_2)$ is used. Unigrams and trigrams are updated accordingly.

The tagger is evaluated on English and German data (Penn-Treebank (Marcus et al. (1993) and Negra (Skut et al. (1997)) corpora, correspondingly). The results of the experiments on both corpora amount to 96.7%.

Maximum Entropy

Maximum entropy modeling is a framework for combining statistical evidence from heterogeneous sources for classification. The basic idea of the framework consists in choosing the distribution with the highest possible entropy such that it satisfies all the constraints given by the present evidence. Maximizing the entropy aims at preserving as much uncertainty as possible, which inherently serves as regularization to avoid overfitting.

The Maximum Entropy Tagger (MXPOST) described by Ratnaparkhi (1996) is a classification-based tagger based on a maximum entropy probability model. The tagger assigns a word to one of the pre-defined classes (parts of speech). The classification function implemented with maximum entropy probability models maximizes the joint probability of a tag `a` and a context `b` based on binary *features* that are "active" for the pair `(a, b)` (i.e. are equal 1). Formally, this probability `p(a|b)` is defined as

$$p(a|b) = \pi\mu \prod_{j=1}^{k} {\alpha_j}^{f_j(a,b)}$$

where π is a normalization constant, $\{\mu, \alpha_1, \ldots, \alpha_k\}$ are the positive model parameters and each parameter α_j corresponds to a feature f_j. Moreover, given a sequence of words $\{w_i, \ldots, w_n\}$ and tags $\{a_i, \ldots, a_n\}$ as training data, b_i is defined as a context available when predicting a_i.

Features encode the information which can help the model to correctly identify the class of a token. They can refer to the tokens and tags in the context, to the lexical characteristics of the token being tagged or any other information that contributes to correct assignment of a tag. For example, the following feature is active if the word ends with "*-ing*" and is tagged as a verb gerund:[7]

$$f_j(a,b) = \begin{cases} 1 & \text{if suffix(w}_i) = \text{"}\texttt{ing}\text{" \& a} = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

[7]Example is taken from Ratnaparkhi (1996).

The corresponding model parameter α_j will contribute to the joint probability `p(a|b)` if the conditions are satisfied.

The model is provided with a set of feature templates which incorporate conditions on (a) the current word, the following and preceding two words and two tags, for frequent words and (b) the first and last four letters of the current word, as well as the information whether the current word contains numbers, uppercase characters and hyphens, for rare words. The instantiations for variables in the feature templates (tags, words, suffixes, etc.) are obtained automatically from the training data. The model parameters for the distribution p are obtained via *Generalized Iterative Scaling* described by Darroch and Ratcliff (1972). Once trained, the model tags tokens by choosing a tag with the highest probability.

The tagger was trained and tested on the Penn-Treebank corpus. It demonstrated the state-of-the-art performance, achieving an accuracy of 96.6%.

Hajič and Hladka (1998) applied the tagger for morpho-syntactic annotation of Czech, a highly inflective language whose tagset size exceeds three thousand tags. The experiments show a significant improvement of results over an HMM-based n-gram model and provide an accuracy of 93.8%.

2.3.2 Symbolic Machine Learning

Memory-Based Learning

Memory-based learning is a type of supervised, inductive learning. It is also known as *similarity-based, example-based, instance-based, case-based* or *exemplar-based* learning. In this learning approach, examples from the training data are stored in the memory without modification and new examples are classified by comparing them to previously seen instances. In this regard memory-based learning is a type of *lazy* learning: no abstraction is made over the training data.

Examples provided for a memory-based learning model are represented as a vector of feature values. One of the features stands for the category to which the example is assigned. This feature has an empty value in test examples.

When a test example is provided to the model for classification, relevant instances are extracted from the memory. To avoid errors caused by the noise in the training data, each test example is compared to several instances from the memory base. The classification is, thus, based on the k-nearest neighbor algorithm. The relevant instances are extracted from the memory

$$distance(X,Y) = \sum_{i=1}^{n} \delta(x_i, y_i)$$

$$\delta(x_i, y_i) = \begin{cases} \frac{|x_i - y_i|}{max_i - min_i} & \text{if the values are numeric, else} \\ 0 & \text{if } \mathrm{x}_i = \mathrm{y}_i \\ 1 & \text{if } \mathrm{x}_i \neq \mathrm{y}_i \end{cases}$$

Table 2.3: *Overlap metric*: equations for instance and feature distances.

$$distance(X,Y) = \sqrt{\sum_{i=1}^{n} \delta(x_i, y_i)}$$

$$\delta(x_i, y_i) = \begin{cases} (\mathrm{x}_i - \mathrm{y}_i)^2 & \text{if the values are numeric, else} \\ 0 & \text{if } \mathrm{x}_i = \mathrm{y}_i \\ 1 & \text{if } \mathrm{x}_i \neq \mathrm{y}_i \end{cases}$$

Table 2.4: *Euclidean distance*: equations for instance and feature distances.

on the basis of a similarity metric. The performance of the model crucially depends on the choice of a similarity metric. Two basic metrics are generally used: the *overlap metric* and the *Euclidean distance*. Both of them compute the similarity (distance) between instances X and Y based on the similarity (distance) of pairs of corresponding features of the examples. In the *overlap metric*, the contributions of the feature distances of all pairs is summed up. *Euclidean distance* is a square root of the sum of the feature distances. The metrics also differ in the way the distance between numeric features is calculated. The equations for the instance and feature distances for both metrics are presented in Tables 2.3 and 2.4.

Features should be provided to the model by an experimenter. Since features represent the basis for classification, the performance of a memory-based model to an extreme degree depends on the feature selection. Weighting of the features helps moderating the problem by introducing a bias towards essential features and discounting irrelevant and redundant features. Many feature weighting methods are described in the literature. Among them, *Information Gain* (Quinlan (1986)), *Gain Ratio* (Quinlan (1993)), *Chi-squared* weighting and its normalized version *Shared Variance* (White and Liu (1994)).

Memory-based learning has been successfully applied to tagging (Daelemans, Zavrel, Berck and Gillis (1996), Cardie (1993)). Features in memory-based taggers usually refer to the focus word itself, its morphology and a local context of two-three neighboring words. Words are assigned one of the pre-defined POS or morpho-syntactic classes.

A tagger described by Daelemans, Zavrel, Berck and Gillis (1996) is one of the most well-known memory-based taggers. It is part of the TiMBL system, developed at Tilburg University. One of the novelties of the tagger is the employment of the IGTree algorithm to compress the memory base into trees, which allows for optimized retrieval speeds (100 to 200 times faster than normal memory-based retrieval) and the memory storage of the model (over 95% less memory). Instances in the model contain information about (a) the focus word, the preceding and following word forms, the two preceding (already disambiguated) tags and the one following (still ambiguous) tag for known words and (b) the first letter and the three last letters of the focus word, the preceding and following word forms, the one preceding (already disambiguated) and one following (still ambiguous) tags for unknown words. *Information Gain* is used for feature weighting. Instance distance is calculated with the weighted *overlap metric*:

$$distance(X,Y) = \sum_{i=1}^{n} \omega_i \delta(x_i, y_i)$$

where ω is a weighting function.

The tagger has been trained and evaluated on the Penn-Treebank corpus and has achieved the state-of-the-art performance of 96.4%. The tagger has also been used in the experiments with POS-tagging of Dutch, a language close in its characteristics to German. The experiments are described by Daelemans, Zavrel and Berck (1996). In the experiments, the tagset was restricted to 13 main categories. The results shown in the experiments (95.7%) are reported to be as good or better than the results achieved by state-of-the-art rule-based and statistical approaches to POS-tagging of Dutch.

Decision Trees

Decision tree learning is one of the most widely used methods for inductive inference. It is based on the use of decision trees for classification tasks. A decision tree represents a predictive model which incorporates observations about an object in terms of attribute-values pairs and allows for the inducing of the target class of the object from these observations. An example decision tree is shown in Figure 2.4. The tree makes decisions about a tagger based on its characteristics and identifies it as one of the following taggers: Transformation-Based Tagger (TBT), English Constraint Grammar Tagger

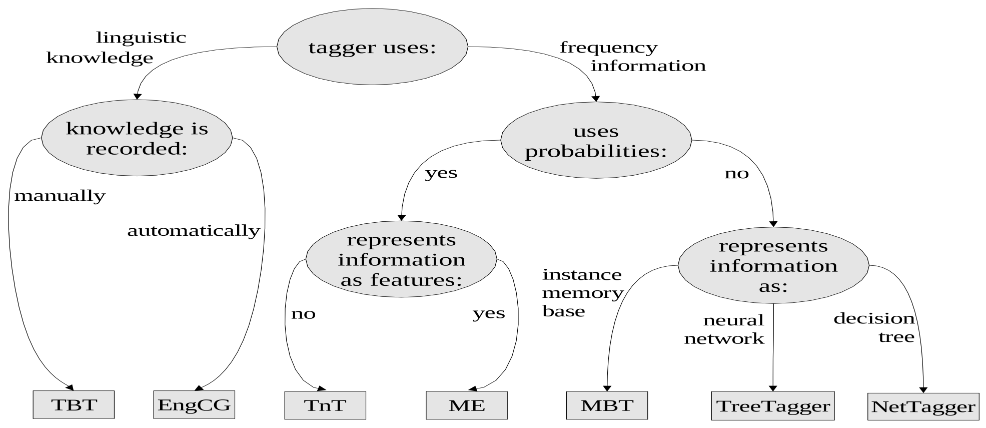


Figure 2.4: An example of a decision tree.

(EngCG), Trigrams'n'Tags Tagger (TnT), Maximum Entropy Tagger (ME), Memory-Based Tagger (MBT), Decision Tree Tagger (TreeTagger) or Neural Network Tagger (NetTagger).

Each node in a tree specifies a test on some attribute of the object, whereas the branches leading from the node correspond to possible values of the attribute. Leaves of a tree are instantiated as elements of a discrete set of possible categories to which the object can be assigned. The object is classified by starting at the root node, testing its question, following the branch with the appropriate answer to a daughter node and repeating the process until a leaf node is reached.

Decision trees are built by at first constructing a large tree and then *pruning* it. Most algorithms used for construction of a decision tree are based on a core algorithm that can be exemplified by the ID3 algorithm (Quinlan (1983)). With this algorithm, the tree is constructed recursively from the training examples. At each recursion step, an attribute is found that splits the examples into maximally distinct groups. At first, this is done for the whole set of examples. The attribute is encoded as a root node, descending branches correspond to the values of the attribute. The procedure is repeated for each new node until all the training examples are classified.

The attribute that splits the examples best is found by maximizing the *information gain* – an information-theoretic metric defined as the difference of the entropy of the mother node and the weighted sum of the entropies of the child nodes:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{| S_v |}{| S |} Entropy(S_v)$$

where $Values(A)$ is the set of all possible values for the attribute A, and S_v is the subset of S for which attribute A has value v.

After a tree has been constructed, it is *pruned.* Pruning is the process of removing a subtree rooted at a node and making the node a leaf with the category that is most common for the leaves dominated by the node. Pruning allows for avoidance of overfitting which results from basing decisions on coincidental regularities in the training data.

The application of decision tree learning in tagging have been explored by many researchers: Schmid (1994*b*), Magerman (1995), Márquez (1999), Orphanos et al. (1999). Magerman (1995) uses decision trees for a number of simultaneous decision-making problems, such as assigning POS labels to words, determining the constituent boundaries in a sentence, determining the constituent labels, etc. The reported results on Penn-Treebank data (96.6%) demonstrate the high competitiveness of the approach.

Orphanos et al. (1999) apply the approach to POS-tagging of Modern Greek. The POS disambiguator is built as a "forest" of decision trees for each ambiguity class in Modern Greek. For each ambiguous token in a text, its ambiguity class is determined and the corresponding decision tree is selected. Tests in the trees represent conditions on the context. Typical tests are the POS label of the previous token, the gender value of the next token, etc. Traversing the tree returns the contextually appropriate POS label for the token being tagged. The reported accuracy ranges between 92.52% and 95.19%.

Neural Networks

A rather new approach to tagging is based on artificial neural networks. A neural network is a collection of simple processing units. The units are organized in layers and weighted connections exist between all units in the adjacent layers. The information is fed to the units of the bottom layer, called an *input layer.* Each unit then passes its given value through the connections to the units on the next layer. A value received by a unit on the next layer, called an *activation value*, is computed as a sum of values from the incoming connections multiplied by a weight number of the connection, with a threshold value subtracted. A simple computation is then performed on the received value in order to restrict the value to a certain range. Usually, the *sigmoid* function is used to map the values into the interval [0,1]:

$$a_{ij} = \frac{1}{1 + e^{-net_j}}$$

where net_j is a received value and a_{ij} is a resulting output value.

The output value is then further passed through the connections leading out of the unit to the units of the next layer. This process is repeated until the top layer, called an *output layer*, is reached. Initially, the weights on connections are set to small random values. The network *learns* by adapting the connection weights and thresholds so that the correct output is produced. The *backpropagation* algorithm is usually used for this adaptation.

When artificial neural networks are applied to tagging, the input information represents possible tags for a given token and the information about the context of the token. Each unit of the output layer corresponds to one of the tags of the tagset. During the training phase, the network learns to activate the output unit that corresponds to the correct tag of the token and to deactivate all other output units. In the trained network, the output unit with the highest activation value corresponds to the tag for a currently processed token.

The taggers based on artificial neural networks differ in respect to the type of the network used. Taggers based on networks with different number of intermediate (*hidden*) layers were reported in the literature (Schmid (1994*a*), Marques and Lopes (1996)), as well as taggers based on recurrent networks (Pérez-Ortiz and Forcada (2001)) and tagging systems that combine several networks (Ma and Isahara (1997)).

Schmid (1994*a*) describes a tagger based on one of the simplest kinds of networks, a two-layer perceptron. This network does not have any hidden layers: the inputs are fed directly to the outputs via a series of weights. The input to the network represents probability distributions of all tags for the token being tagged and for three preceding and two following tokens. The tagger was trained and evaluated on the Penn-Treebank corpus. The performance of the tagger was compared to the performance of a trigram based tagger (Kempe (1993)) and a Hidden Markov Model tagger (Cutting et al. (1992)) which were trained and evaluated on the same data. The neural network-based tagger is reported to perform as well as the trigram-based tagger and better than the HMM-tagger, achieving the accuracy of 96.22%. Additional experiments were performed to estimate the influence of the size of the input context and of the additional hidden layer in the network. The additional layer turned out to deteriorate the performance,

EngCG	Brill	TnT	ME	MBT	TreeT	NetT
99.7%	96.3%	**96.7**%	96.6%	96.4%	96.6%	96.2%

Table 2.5: Performance of the taggers on Penn-Treebank data.

as well as restricting the context to two preceding and one following words. Increasing the context window did not lead to any improvement.

2.4 Comparison of the approaches

Many characteristics can serve as a basis for comparison of the approaches. The most obvious and the most important aspect concerns tagging performance. Another crucial question is whether and to which extent a tagging method relies on linguistic knowledge and existing corpora. This is particularly important for less thoroughly explored languages and for languages with limited data resources. For languages which have been in a research focus for an extended period of time, such as English, technical characteristics of taggers are of a bigger concern. Thus, for English, most taggers show a similar high performance of 95-97% and the value of taggers is also evaluated on their complexity, speed and storage requirements. For morpho-syntactic annotation, the sensitivity of a tagger to the size of the tagset is one of the most important features. Since incorporation of the morphological information into the tagset leads to a significantly explosion of the tagset, performance of a tagger can drastically differ when the tagger is applied to the tasks of pure POS annotation and of morpho-syntactic tagging. Context sensitivity is another characteristic important for the task of morpho-syntactic annotation, since clues necessary for successful morpho-syntactic disambiguation are often of non-local nature. A comparison of the approaches based on these aspects is presented below.

2.4.1 Performance

All the taggers described above have been evaluated on the Penn-Treebank data, which partially simplifies the comparison of their performance.[8] Table 2.5 summarizes results achieved by the taggers reported in correspond-

[8]Basing experiments on the same corpus does not eliminate such complicating factors affecting performance comparison as size of the training data, smoothing method used and quality of the guesser.

ing publications.[9] The best performance which significantly exceeds performance of any other tagger is demonstrated by the hand-crafted rule-based EngCG tagger. However, a comparison of the results of the EngCG tagger and of other taggers is complicated by the fact that the EngCG tagger leaves an ambiguity on 3-7% of tokens. At the same time, Schmid (1994*a*) mentions that the accuracy of the NetTagger can be raised to 97.69% if 4.6% of tokens are left ambiguous. Since remaining ambiguity corresponds to keeping more than one tag, such relieving of conditions raises a chance of correct tagging for the other methods as well.

A relevant experiment comparing the performance of the EngCG tagger to the performance of a trigram tagger on the same data is described by Samuelsson and Voutilainen (1997). The error rate of both taggers has been presented as a function of remaining ambiguity on tokens. For all values of the tags/token correspondence in the reported range of 1.026-1.093, the error rate of the EngCG tagger is several times lower.

A similar experiment has been performed for French by Chanod and Tapanainen (1995). The experiment compares a hand-crafted rule-based tagger and an (unsupervised) HMM tagger. Full disambiguation has been required for both taggers. Additionally, a time limit on amount of time spent on development and tuning of both taggers (one man-month) has been imposed. This time constraint has been designed to empirically check the general believe that development of rule-based taggers is much more time-consuming than tuning of statistical taggers. The results achieved by Chanod and Tapanainen (1995) show that even with the limited time spent on rule development, the rule-based tagger has yielded an accuracy of 97.5% which corresponds to an improvement of 2.5% over the accuracy of the HMM tagger.

These experiments provide more evidence supporting the statement that hand-crafted rule-based taggers outperform in accuracy the best taggers of the competing approaches, namely probabilistic taggers.

All the taggers included in Table 2.5 are supervised taggers. Merialdo (1994) presents experiments aimed at the comparison of supervised and unsupervised probabilistic taggers. The experiments have shown that supervised taggers outperform unsupervised taggers and the quality of the performance is directly connected to the amount of hand-tagged training data available. Training on untagged data improves the performance if only

[9]Taggers described in Table 2.5 are: English Constraint Grammar Tagger (EngCG), Brill Tagger (Brill), Trigrams'n'Tags (TnT), Maximum Entropy Tagger (ME), Memory-Based Tagger (MBT), decision tree based tagger (TreeT) described by Magerman (1995) and a neural network tagger (NetT) described by Schmid (1994*a*).

a very restricted pre-tagged corpus is available.

Megyesi (2001) describes experiments which aim at the comparison of the performance of four of the data-driven taggers described above, namely, the Memory-Based tagger (MBT), the Maximum Entropy tagger (ME), the Brill tagger and the Trigrams'n'Tags tagger (TnT), on Swedish data. From a syntactic and morphological point of view, Swedish can be placed between English and German, as it is characterized by a richer morphology than the morphology of English and by a relatively flexible word order. The findings of Megyesi (2001) demonstrate the same performance distribution between the taggers as the distribution shown on English data. The spread in accuracy is more distinct, though: 4.49% (89.06% – 93.55%) as compared to a spread of 0.4% in experiments with the English data, which can be explained by the characteristics of Swedish which make the language more difficult for tagging and by a bigger size of the tagset (139 tags as compared to the 36 tags of the Penn-Treebank data).

2.4.2 Need for linguistic knowledge and corpora

A choice of a tagging method is often guided by the availability of resources for the focus language. Supervised methods cannot be applied unless enough annotated data are accessible for the language. In this regard unsupervised approaches represent a suitable alternative, since they do not rely on pre-tagged training data. Most data-driven tagging approaches provide both supervised and unsupervised versions for tagging. The most well-known and widely used unsupervised taggers are HMM taggers and unsupervised transformation-based taggers. However, Merialdo (1994) has demonstrated that unsupervised taggers yield a lower performance compared to supervised versions.

Supervised approaches can be compared on the basis of their performance on restricted training data sets. Such an experiment has been performed by Megyesi (2001) for the Memory-Based tagger, the Maximum Entropy tagger, the Brill tagger and the Trigrams'n'Tags tagger (TnT), on Swedish data. The taggers were trained on different portions of data in the range of one thousand to one million tokens. The Brill tagger has demonstrated the smallest spread in the error rate (50% error rate reduction) and the Maximum Entropy tagger the largest spread (88%). The overall result has shown that TnT provides the best results when the training data contain more than ten thousand tokens, otherwise the Brill tagger outperforms the three other taggers. Additionally, experiments of Schmid (1994*a*) and Schmid and Kempe (1996) have demonstrated that the NetTagger outperforms a trigram tagger (Kempe (1993)) on small amounts of training data (up to approximately 80 thousand tokens).

A tagging approach that does not require training data and provides a high accuracy is the hand-crafted rule-based approach. Creation of a successful hand-crafted rule-based tagger relies, however, on linguistic knowledge introduced to the system by a linguist. Although this hand-crafted approach is regarded as the most dependent on manually incorporated linguistic knowledge, for feature-based approaches, such as memory-based learning and maximum entropy framework, such knowledge is also required in the feature selection process.

2.4.3 Technical characteristics

Technical characteristics of taggers mostly concern the speed of training and testing and the complexity of taggers, often seen as storage requirements.

Rule-based methods incorporate information in the form of constraints which are usually simple and small. Together with the decision trees they represent the most compact systems. Rule-based methods are additionally easy to check and correct, which makes them attractive from the maintenance and tuning point of view. Probabilistic taggers incorporate large numbers of lexical and contextual probabilities, which makes them more massive than the rule-based taggers. The highest storage requirements are imposed by memory-based taggers, since all the training instances are kept in the memory.

The complexity of the taggers is, however, not reflected in their speed. Megyesi (2001) reports a speed required for learning 100 thousand tokens as approximately one second for TnT, a minute for the Memory-Based tagger, and one day for the Brill tagger and the Maximum Entropy tagger. Tagging the same amount of data takes the same time as for training in case of TnT and the Memory-Based tagger, and is approximately as fast for the Brill tagger as for the Memory-Based tagger.

The corresponding numbers for other taggers are not available.

2.4.4 Sensitivity to the size of the tagset

This characteristic is particularly important for the task of morpho-syntactic tagging, since the morphological information, such as *case, number* and *gender*, has to be included into the tagset. This complicates the tagging task, since the system has to choose a correct analysis from a much bigger set. As numerous experiments described in the literature show (Wothke et al.

EngCG	Brill	TnT	ME	MBT	TreeT	NetT
unrestr.	$\leq$ 3(f+p)	2p	$\leq$ 2p+2f	2p+1f	$\leq$ 2p+2f	3p+2f

Table 2.6: Context window utilized by the taggers.

(1993), Steiner (1995), Hajič and Hladka (1997), Tufiş (2000)), expansion of the tagset leads to a significant decrease in performance. The experiments of Megyesi (2001) have demonstrated that TnT is the least sensitive to the tagset expansion from the four taggers which were explored. Adding more training data can help to lower the error rate.

Hand-crafted rule-based taggers also suffer, though to a lesser degree, from expanding the tagset. More fine-grained category distinction requires a larger set of constraints and full disambiguation is harder to achieve with big tagsets. Thus, for hand-crafted rule-based taggers, tagset expansion concerns first of all a decrease in recall.

2.4.5 Context sensitivity

For POS tagging, most taggers utilize a context window of 2-6 surrounding tokens for successful realization of the task. Table 2.6 summarizes context window values for the taggers described in sections 2 and 3. Letters `p` and `f` stand for preceding and following tokens, correspondingly. Thus, "`2p + 1f`" should be read as "2 tokens in the left and 1 token in the right contexts" and "$\leq$ `3(f+p)`" as "up to 3 tokens in the surrounding context".

Constraints of the EngCG tagger are in 71% cases non-local (Voutilainen (1995*b*)), i.e. they incorporate a context extending beyond the neighboring words. The rule formalism does not restrict the context to any particular length and, since the tagger does not require training, the development speed is independent of the context window used. However, the such non-locality of the rules can lead to a lower annotating speed.

The TnT, the Memory-Based Tagger and the NetTagger incorporate the context as a unit, i.e. it cannot be broken into features, as it is done in the Brill tagger, the Maximum Entropy tagger and in the Magerman's decision tree tagger, and has to be processed together (e.g., a bigram `ART NOUN`). Thus, including more tokens in the context would increase the *data sparseness* problem: longer sequences of tags occur in the data less often than short sequences and the tagger would suffer from insufficient training data, which would result in a decrease in tagging accuracy.

For the Brill tagger, the Maximum Entropy tagger and the Decision

Tree tagger, the problem is less severe, since decisions are made on the basis of distinct features, such as "a tag of a second token in the right context". However, expanding the context results in significant slow-down of the training phase of these taggers, since the feature space is considerably increased.

2.4.6 Summary of the comparison

Different tagging approaches and taggers have been compared on the basis of different characteristics. Which of the approaches and taggers are more suitable depends to a high degree on the intended application, the amount of time set up for development, tuning and training of the tagger and on the availability of resources for the focus language. If a sufficient amount of pre-tagged training data is available and the need for a tagger is urgent, n-gram models provide the best alternative, since they can be trained in several minutes and since they achieve a high accuracy. However, n-gram taggers are less suited for the tasks which require taking a broad context into account for the successful realization. Maximum entropy taggers incorporate information in distinct features and are able to capture broad context in a better way. They, however, are more time-consuming and require large amounts of data for training. The development of a hand-crafted rule-based tagger is the optimal solution if no or only very restricted training data exist and high accuracy is demanded. They also are best suited for disambiguation which relies on a broad context. These characteristics make them an ideal candidate for morpho-syntactic annotation of German, given that large annotated corpora are not accessible for this language.

A common obstacle for successful tagging with all the existing methods is a significant expansion of a tagset. For approaches that produce only one tag for each token, such an expansion results in a lower accuracy. If remaining ambiguity is allowed, adding information to the tagset leads to a lower recall. Combining techniques of the approaches and extending the methods have been presented in the literature as a means for the improvement of tagging accuracy. Such hybrid and modified methods are discussed in the next section.

2.5 Hybrid methods and extensions to the approaches

2.5.1 Combining rule-based and probabilistic methods

Hajič et al. (2001) describe a hybrid rule-based and statistical tagging system for Czech. The combination of the components is serial and aims at bringing together the strengths of hand-crafted rule-writing and of probabilistic learning: the rule-based module performs initial disambiguation of the text. It aims at keeping the recall close to 100% and at accurate reduction of the search space for the subsequent application of a probabilistic component. The probabilistic component is based on a trigram model. It resolves remaining ambiguity by choosing the most likely analysis given the context. The final system outperforms other Czech taggers described in the literature and reaches an accuracy of 95.38%.

A similar hybrid tagging system was successfully designed for English Tapanainen and Voutilainen (1994). The obtained results amount to 98.5% accuracy.

2.5.2 Combining n-grams and decision trees

Schmid (1994*b*) has used a decision tree formalism for estimating the transition probabilities in n-gram taggers. A decision tree incorporates tests on the context and the probability of an n-gram is determined by following a corresponding path in the tree. Leaf nodes of the tree contain probability distributions for different tags.

The TreeTagger was originally designed for English (Schmid (1994*b*)). The performance of the tagger was tested on data from the Penn-Treebank corpus. An achieved accuracy of 96.36% demonstrates a small improvement over the results received by a trigram tagger on the same data (96.06%). Applying the tagger to German demonstrated less satisfying results due to the smaller amounts of training data available (Schmid (1995) reports that about 50% of tokens in the test data did not occur in the training data). An extended version of the tagger which aimed at improving poor lexical probability estimates has been developed and described in Schmid (1995). The extensions concerned smoothing lexical probabilities with equivalence class based probabilities and a more sophisticated treatment of sentence-initial words. Further improvement was obtained by incorporating a prefix lexicon and expanding the full form lexicon. The performance of the new version of the TreeTagger yielded a 37% improvement over the first version and achieved an accuracy of 97.5%.

The TreeTagger is robust with respect to the size of the training data. Its accuracy deteriorates slower than the accuracy of a trigram tagger when the amount of training data is being reduced. The processing speed of the TreeTagger is also higher than the speed of a trigram tagger.

2.5.3 Reduced tagset approaches

A very promising approach to deal with issues of data-sparseness for large tagsets has been suggested by Tufiş (2000) and Tufiş et al. (2000). They have advocated the methodology of Tiered Tagging with Combined Language Models (TT-CLAM).

Central to the TT-CLAM approach is an algorithm for automatically reducing large tagsets into a hidden tagset that is used for training the language model for POS tagging proper and that is manageable in size for current tagging technology. This hidden tagset is designed in such a way that the full tagset can be recovered almost deterministically on the basis of lexical information associated with a given token. The words that become ambiguous after mapping reduced tags back to the full tagset (less than 10% in the experiments of Tufiş) are further disambiguated by a small set of contextual rules. TT-CLAM has been successfully applied to a number of languages, including Romanian and Hungarian. For Romanian Tufiş (2000) reports a tagging accuracy of between 97% and 99% and a mapping accuracy of almost 99% when the hidden tagset is mapped back to the full tagset.

A similar approach based on a reduced tagset is described by Dienes and Oravecz (2000) for Hungarian. The two approaches differ in the methods used for tagset compaction. The TT-CLAM approach of Tufiş (2000) successively reduces the size of the original tagset (allowing for a 10% loss of information) while the bottom-up tagset design of Dienes and Oravecz (2000) maximally reduces the original tagset without loss of information and then re-introduces morpho-syntactic features to expand the reduced tagset to a set of tags that exhibit sufficient distributional cues for the tagger.

2.6 State of the art in tagging of German

Numerous taggers have been reported in the literature for German. The best results for pure POS tagging are achieved by the TreeTagger described by Schmid (1995) (97.50%) and by the TnT tagger (96.7%). These results are comparable to the performance of the state of the art taggers applied to

	IBM	Münster	Stutt-gart	Stutt-gart	Stutt-gart	Pader-born
method	bi- & tri-grams	bi-grams	bi-grams	tri-grams	decision trees	tri-grams
training data	20 k tokens	200 k tokens	18–22 k tokens	18–22 k tokens	18–22 k tokens	20.000 tokens
tagsets large/small	689 33	143 54	855 51	855 51	855 51	456 51
accuracy large/small	77.70 93.40	81.50 92.80	33.81 85.38	64.91 87.18	58.98 95.14	**84.70** **95.90**

Table 2.7: Comparison of German Taggers.

annotation of English data. The morpho-syntactic tagging of German has also been explored. However, for this task less success has been achieved.

Table 2.7 describes German taggers which have been applied to morpho-syntactic annotation (tagging with the expanded tagsets) along with pure POS tagging. The taggers included in the table have been presented by Wothke et al. (1993) (IBM), Steiner (1995) (Münster), Schmid and Kempe (1996) (Stuttgart) and Lezius et al. (1996) (Paderborn). As Table 2.7 shows, the best results are yielded with a trigram model developed by Lezius et al. (1996). An achieved accuracy of 84.70% is lower than the results reported for probabilistic morpho-syntactic annotation of other languages: compare, for example, with an accuracy of 93% reported for Czech by Hajič and Hladka (1998). These results are also considerably lower than the results of hybrid and extended methods: cp. 95.16% for Czech (Hajič et al. (2001)) and 97%–99% for Romanian (Tufiş (2000)).

The reason for such difference in performance on the data of different languages lies in a higher ambiguity of German data, a more intricate case syncretism and a relatively free word order of German. Alternative methods for successful realization of the task of morpho-syntactic annotation of German are explored in Chapter 6.

2.7 Conclusion

In this chapter, the standard approaches to tagging have been presented. A detailed description of different techniques together with the survey of some of the well-known modern taggers for each approach has been provided. The described taggers have been compared on the basis of different parameters and their applicability to the task of morpho-syntactic tagging has been highlighted. Additionally, hybrid and extended systems relevant for the task have been described. The state of the art of German tagging presented in the chapter demonstrated a need for new methods for successful realization of the task of morpho-syntactic annotation of German. Such methods are explored in Chapter 6.

Chapter 3

Dependency Parsing

One of the possible areas of application for morpho-syntactic taggers is preprocessing of data for dependency parsing. Providing unambiguous morpho-syntactic information in the input of a dependency parser considerably simplifies the task of the parser, since in many languages the grammatical functions of tokens are closely connected to their morphological values. Thus, Reis (1982) explores the status of a category of a subject in German and argues that this category is irrelevant for the language in question, i.e. there is no language regularity that cannot be described without reference to the notion of subject if a simpler notion of a nominative NP is available. This claim of Reis (1982) supports the argument of importance of morphological information, in particular morphological case, in syntactic analysis.

Dependency analysis has a long history in linguistic theory and lately, a renewed interest has been shown to dependency analysis in theoretical, computational and corpus linguistics: new theories, parsing models and treebanks based on the dependency framework are being developed and explored.

Dependency theory has been particularly popular in research of languages with flexible word order, such as Czech and Russian, since it provides the means for a more elegant modeling of word order variations than the constituency analysis. In German linguistics, dependency theory received broad support soon after the publication of the first detailed description of the theory presented by Tesnière (1959): studies of Heringer (1970), Engel (1972) and Kunze (1975) have established rich traditions of German dependency syntax.

The current chapter aims at providing the theoretical background for the application of morpho-syntactic information to dependency parsing that will be described in Chapter 7. This chapter is organized as follows: after short preliminaries in section 3.1, section 3.2 introduces the classical model of dependency theory by sketching the basic notions and principles shared by most current dependency frameworks. The introduction is based on the work of the originator of the theory, Lucien Tesnière (Tesnière (1959)), as well as on the work of Mel'čuk (1988) and on course materials of Hudson (2000) and Kruijff and Duchier (2002). In Section 3.3, the introduction is followed by a discussion of various issues of dependency theory which represent points of divergence among current dependency frameworks. Section 3.4 examines the role of dependency theory in computational linguistics in and Section 3.5 provides a survey of current German dependency parsing models. The chapter is concluded in section 3.6.

3.1 Preliminaries

The theory of dependency analysis represents one of the two major theories of linguistic analysis. Unlike the competing theory of *constituency*, or *phrase-structure*, analysis, which is a relatively recent development[1], dependency theory can be traced to the ancient Greek and Indian linguistic traditions (Fraser (1994)).

The first linguistic theory based on dependency has been developed by Tesnière (1953, 1959). It has become a classical model of dependency theory and has since been extended to numerous theories, just to mention a few, Case Grammar (Anderson (1971)), Constraint Dependency Grammar (Maruyama (1990)), Functional Dependency Grammar (Tapanainen and Järvinen (1997)), Functional-Generative Description (Sgall et al. (1986)), Meaning-Text Model (Mel'čuk (1988)), Word Grammar (Hudson (1984)).

Below, a classical model is outlined, and current issues in dependency theory are discussed.

3.2 Classical Model

3.2.1 Basic notions

The theory is based on the notion of *dependency* – a binary asymmetric, irreflexive, transitive relation between linguistic units. Dependencies can be established on different levels of linguistic analysis. Figure 3.1 exemplifies morphological, semantic and syntactic dependencies for sentence (7).

[1]It was first introduced in early 1930s (Bloomfield (1933)).

For example, on morphological level, a dependency between "*Mann*" and "*dieser*" is established, since the morphological characteristics of the token "*dieser*", such as its gender and number, are dictated by the morphological characteristics of the noun.

(7) Dieser Mann hat eine schöne Frau.
'This man has a beautiful wife.'

Since the emphasis of the current chapter is on dependency parsing, the exposition below is restricted to the syntactic level of dependency analysis.

Elements connected by a dependency are called a *head* and a *dependent*.[2] Several criteria have been proposed in the literature for the identification of the head of a dependency. Fraser (1994) distinguishes the following criteria:

- the head determines whether a dependent is optional or obligatory, and not vice versa;
- the head subcategorizes for its dependents, and not vice versa;
- the head determines in which inflectional form of a dependent occurs, and not vice versa;
- the head identifies the semantic object which a dependent further specifies, and not vice versa.

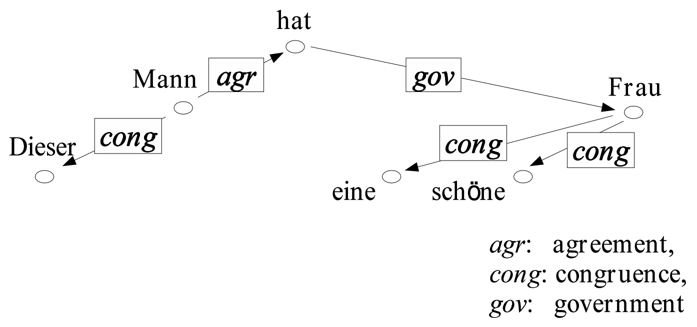
Dependency relations between elements of a linguistic object, such as words in a sentence or morphemes in a word, constitute a *dependency graph*. The classical model further restricts dependency graphs to trees by imposing the following constraints:

- only one node in a graph is independent;
- all other nodes have a head;
- each node has at most one head;
- head and dependent are adjacent.

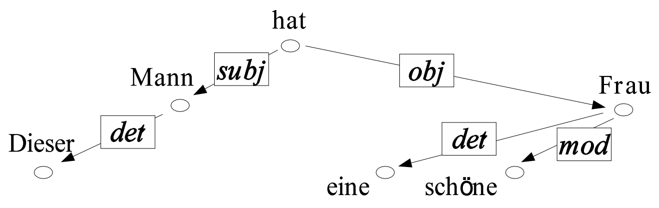
According to Tesnière (1959), the basic syntactic unit which corresponds to a node in a dependency tree is a *nucleus*. A nucleus always contains both the structural and the semantic centers and consists of one or more, possibly

[2] Other terms used in the literature are: *governor* or *regent*, for the head, and *modifier*, for the dependent.

Morphology:



Syntax:



Semantics:

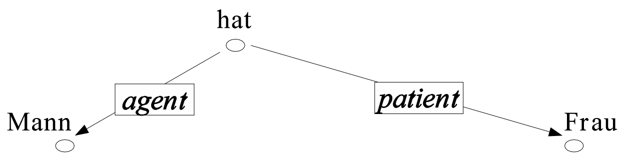


Figure 3.1: Different types of dependencies

discontinuous, words or parts of words. An example of a nucleus in English is a verbal phrase which consists of an auxiliary verb and a participle, such as "*be eating*", "*have eaten*". The syntactic center of such a nucleus is an auxiliary verb, whereas the semantic center is contained in a participle.

An ordered set of nodes of a complete subtree of a particular node represents a *projection* of the node. An important property of dependency trees is based on the notion of projection: a dependency tree is said to have a property of *projectivity*, or is called *projective*, if projections of all nodes of the tree fill continuous intervals.

3.2.2 Comparison of dependency analysis to constituency analysis

	Constituency	**Dependency**
structure is based on	constituency	relations
main logical operation	set inclusion	establishing binary relations
phrases	explicit	implicit
relations	implicit	explicit
head marking	optional	obligatory
nodes in trees	mostly non-terminal	terminal only
linear order of nodes	obligatory	not required
labeling syntactic roles	no	yes

Table 3.1: Constituency analysis vs. dependency analysis

Table 3.1 summarizes the main differences between dependency and constituency analyses.

Dependency structures are based on relations between linguistic units, i.e. the structures show which units relate to each other and in which way. Two units are considered to belong together if one is dependent on the other. In constituency analysis, on the other hand, structures reveal how linguistic units combine to form a unit of a higher order. Two units are considered to belong together if they both are a part of a larger whole. The main logical operation of constituency approach is, thus, set inclusion, whereas in dependency approach it is the establishment of binary relations.

Constituency analysis provides explicit identification of phrases as groups of elements. In dependency analysis, phrases are not explicit but can be derived from the structure: a phrase corresponds to a projection of a node

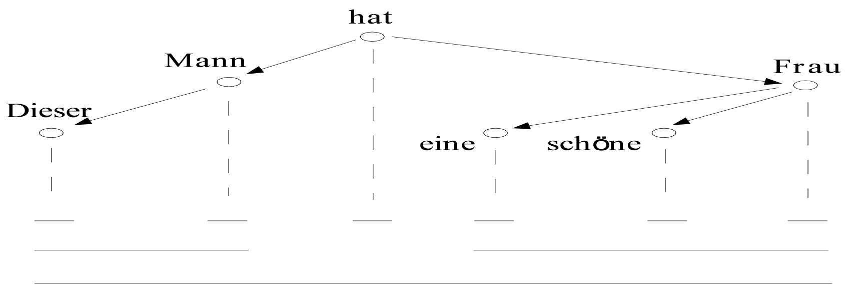


Figure 3.2: Correspondence of a projection of a node to a phrase

together with the complete subtree under the node. In Figure 3.2, projections are presented as horizontal lines under the nodes.

Relations are, on the other hand, explicit in dependency structures but can not be identified in constituency analysis, unless head elements are marked in the constituency structures. Such head marking is optional in constituency theory and is present in only few theories, such as X-bar theory (Jackendoff (1977)) and HPSG (Pollard and Sag (1994)).

Dependency relations are defined directly on linguistic units, so that dependency structures consist of terminal nodes exclusively. Constituency structures are, on the other hand, built of primarily non-terminal nodes. Mel'čuk (1988) provides an example of both types of structures for a sentence which consists of eighteen nodes ("*She loved me for the dangers I had passed, and I loved her that she did pity them*") and shows that while the dependency structure uses exactly the number of nodes equal to the number of words in the sentence (i.e. eighteen nodes), the constituency structure employs sixty one nodes which are filled mostly with non-terminal categorization symbols.

In constituency analysis, the linear order of nodes plays a crucial role. The constituent trees are inseparable from word order. In dependency analysis, linear organization of nodes in a structure is not required. Tesnière (1959) distinguishes between the linear order and a structural order and excludes word order phenomena from his structural consideration. Similarly, Mel'čuk (1988) argues that word order does not belong to the syntactic level but rather is a means of the deep-morphological level. Most modern formal dependency frameworks also separate a level of syntactic dependency analysis from a level of surface ordering of tokens (Gerdes and Kahane (2001), Järvinen and Tapanainen (1998), Bröker (1998)).

In constituency structures, the syntactic roles of units are not specified. In dependency structures, explicit marking of the type of relation between nodes is provided.

3.2.3 Motivation

Two kinds of arguments have been used in the literature for motivation of dependency theory. Arguments of the first kind compare dependency structures to constituency structures and advocate a better suitability of dependencies for the analysis of natural language. The second kind of arguments considers dependency analysis in terms of its plausibility. Below, both kinds of argument are considered.

Advantages of dependency structures as compared to constituency trees can be generally summarized by the following claims:

- dependency structures are more succinct, explicit and expressive;
- dependency structures provide a more elegant means for description of languages with a less fixed word order.

The first claim regards syntactic descriptions of the dependency theory. Unlike constituency trees, dependency trees do not have an intermediate level of non-terminal nodes. Relations are established directly between linguistic units and are marked. At the same time, constituents can be easily derived from the dependency trees. This makes dependency structures more succinct, explicit and expressive in terms of presentation of the information.

The second claim is based on the classical axiom of dependency syntax: dependency trees are invariant with respect to the word order. This axiom makes dependency syntax more attractive for the analysis of languages with a flexible word order, since various linearizations of a sentence can be represented with the same dependency structure. Consider an example of a dependency tree shown in Figure 3.3[3]. This dependency tree corresponds to the following linearizations:

(8) a. Niemand hat diesem Mann das Buch zu lesen versprochen.
b. Diesem Mann hat das Buch niemand zu lesen versprochen.
c. Das Buch zu lesen hat diesem Mann niemand versprochen.
d. Diesem Mann hat niemand versprochen, das Buch zu lesen.
e. Diesem Mann hat, das Buch zu lesen, niemand versprochen.
f. Zu lesen hat diesem Mann das Buch niemand versprochen.

[3]The example is proposed by Gerdes and Kahane (2001).

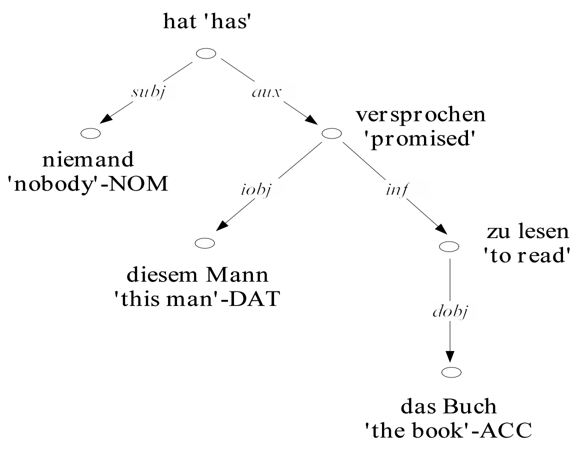


Figure 3.3: Dependency tree of the sentences in (8)

g. Das Buch hat niemand diesem Mann versprochen zu lesen.
'Nobody promised this man to read the book.'

In constituency theory, each sentence in (8) has a separate analysis, although they are syntactically equivalent.[4]

Another advantage of dependency analysis in regard to languages with a variable word order is the unproblematic treatment of discontinuous constituents, such as "*das Buch zu lesen*" in sentence (8g).

The second type of motivation of dependency theory has been presented by Hudson (2003) who argues that dependency structures are plausible from a psychological point of view. The claim is supported by various kinds of evidence which include dependency distance, dependency direction, dependency classification, dependency prototypes, dependency parsing, dependency lexicalization and dependency learning. Thus, Hudson (2003) points out, for example, that dependency structures provide a natural means for estimating the structural difficulty of a sentence calculated in terms of a dependency distance, i.e. the number of words between each word and the word on which it depends.

[4]The sentences convey different information structures as to what is considered focused and what is considered given.

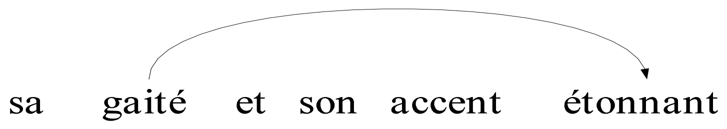


Figure 3.4: Dependency structure provided for a modification construction with coordination

3.2.4 Criticism

Mel'čuk (1988) has addressed criticisms found in literature that concern some of the basic assumptions underlying dependency formalism. The criticisms are related to phenomena of natural language which supposedly cannot be described by the dependency theory. The critiques can be grouped under four headings:

- double dependency: a wordform simultaneously depends on two different wordforms, as in a resultative constructions such as "*wash the dish clean*", where "*clean*" is claimed to depend on both the noun and the verb;
- mutual dependency, such as between a verb and its grammatical subject, where not only the verb governs its subject, but also the subject controls the form of the verb: "*The children are playing*" vs. "*The child is playing*";
- no dependency, as between conjoined items in coordinated phrases, where absence of dependency is proven by a presumed symmetry of coordination: "*John and Mary*" is identical to "*Mary and John*" and both conjuncts have the same weight in the construction;
- insufficient dependency: using dependency syntax, two different structural descriptions cannot be supplied for a modification structure such as "*sa gaieté et son accent étonnant*", since both readings "*his cheerfulness and his astonishing accent*" and "*his astonishing cheerfulness and accent*' are described with the same structure (see Figure 3.4).

Mel'čuk (1988) points out various factors that lead to the inadequacies of criticisms. They include confusion of different types of dependencies (morphological, semantic and syntactic), such as in the case of double and mutual dependencies, as well as in case of "no dependency" criticism. Another factor is ignoring dependency relation names. Attaching a special name to a dependency can prevent the confusion of different readings.

Another natural solution for the problem of dependency insufficiency is introduction of a notion of *grouping*, i.e. treatment of a construction as one unit and separate analysis of the inner and internal dependencies of the construction.

3.2.5 Dependency inventory

There has been no agreement established on a single inventory of dependency labels common for all dependency theories. Three types of dependencies used in different theories can be distinguished according to the level of representation of a theory:

- grammatical functions, such as *subject, object*, etc. (Mel'čuk (1988));
- functional relations, such as *agent, patient*, etc. (one of the levels in the Functional Generative Description of Sgall et al. (1986));
- conceptual relations, such as *agentive, objective*, etc. (Fillmore (1968)).

The tradition of syntactic dependency labeling comes from the syntactic theory of Tesnière (1959), who distinguishes between two types of dependencies: *actants*, which are described as entities participating in the process described by the verb and which correspond to verbal complements, and *circonstants*, which are described as circumstances of the process and which correspond to verbal adjuncts.

Actants discussed by Tesnière (1959) include a *subject*, an *object* and a third actant which is traditionally called *indirect object*, but which does not receive a special name in the theory of Tesnière (1959). Actants are not defined formally, but are described indirectly via their structural meaning, morphological characteristics and examples from different languages.

In the consequent dependency theories concerned with syntax, dependency relations have been usually based on grammatical functions of tokens and include, apart from Tesniere's actants:

- other complements of verbs, such as a *predicative object*;

- relations between verbs in a verbal complex, such as an *infinitival object*;
- adverbial functions, such as a *temporal adjunct*;
- determinative functions, such as a *determiner*;
- modification functions, such as an *attribute*.

3.3 Issues in current dependency theory

The exposition presented above comprises the basic principles of dependency syntax. However, various issues of dependency theory have received different interpretations in modern dependency frameworks. Hudson (1993) discusses areas in which current dependency frameworks deviate from the classical model or disagree among each other. Below, the issues that have influenced the development of the German dependency parser presented in Chapter 7 are briefly stated.

As shown above, constituency structures can be easily derived from dependency structures in any framework. However, the disagreement among different frameworks concerns the **role of constituent structure** in dependency grammar. The opinions range between independent and derivative nature of constituents in the grammar. Thus, Hudson (1976) and Matthews (1981) argue that certain generalizations, such as assignment of mood categories to clauses, can be expressed satisfactory only if reference to constituents is possible. Others (Starosta (1988)) consider phrases strictly redundant, but allow usage of sparse constituency structures in restricted cases. An extreme position of viewing constituent structures as a clear hindrance in grammatical analysis is taken by Sgall et al. (1986) and Hudson (1984).

Treatment of coordinate structures has been a long standing problem in linguistic theory. In dependency paradigms, the difficulty arises from the nature of the structures which lack a head-modifier distinction of elements. As was mentioned above, the introduction of a notion of *grouping* which treats the coordinate structure as one unit has been proposed by Mel'čuk (1988) to resolve the problem. A similar approach to coordination has been advocated by Hudson (1990) who characterizes coordinate structures as *word strings*. The difference between words strings and groupings concerns the internal structure of conjuncts. According to Hudson (1990), conjuncts are organized as (possibly disconnected) dependency structures and each conjunct root holds a dependency relation to some element outside the coordination. Mel'čuk (1988), on the other hand, advocates the analysis according to which one of the conjuncts is the head of the construction and, thus, denies the coordination symmetry at the syntactic level.

Current theories also disagree on **the role of features** in the grammar. In order to incorporate information about subclassification of parts of speech, about valency and inflectional morphology of tokens on nodes, some theories introduce syntactic features (Hudson (1976), Starosta (1988)). This position is not shared by Hudson (1984) who does not allow usage of features and argues that all generalizations should be stated in terms of hierarchy of atomic word-types.

The **treatment of control and raising constructions**, such as "*Fred wants to work*", has recently been one of the questions widely discussed in linguistics. The problematic issue with such constructions is the relation of the noun and the non-finite verb. According to the classical model, each token should have no more than one head, and therefore, the noun cannot depend on the non-finite verb, since it is dependent on the finite verb. Sgall et al. (1986) solve the problem by introducing an empty category which serves as the subject of the non-finite verb and requires coreference with the preceding noun. Other dependency theories, on the other hand, choose to relax the principle that creates the problem and allow tokens to have multiple heads (Anderson (1979), Hudson (1984)).

3.4 Dependency theory in Computational Linguistics

The formal properties of dependency grammar were first investigated by Hays (1964), Gaifman (1965) and Robinson (1970). They have shown that all dependency grammars of the given type have weakly equivalent context-free phrase structure grammars and vice versa. The constraints imposed on the dependency grammar concern dependency structures and the form of dependency rules. The structures are restricted to dependency trees (see the four constraints described in subsection 3.2.1). The rules are required to take one of the following forms:

1) $X(X_1,...,*,...,X_n)$
2) $X(*)$
3) $*(X)$

An element outside a set of brackets represents the head of the construction. In rule type (1), the asterisk marks the position of the head relative to its dependents in the string. In rule type (2), it specifies that `X` occurs without dependents. In rule type (3), it means that `X` does not depend on any other element.

Recently, an increasing attention and interest to dependency theory have been shown in computational linguistics. Dependency parsers have been implemented in different frameworks for various languages. Carroll and Charniak (1992), Eisner (1996) and Yamada and Matsumoto (2003) have developed probability models of dependency grammar for English. Stochastic dependency models have been also presented for Japanese (Matsubara et al. (2002)) and Korean (Chung (2004)). Oflazer (1999) has employed finite-state techniques for dependency parsing of Turkish and Elworthy (1999) has used the same methodology for English. Nivre et al. (2004) have presented a memory-based dependency parser of Swedish. Constraint-based dependency parsers have been proposed for English by Tapanainen and Järvinen (1997) and Maruyama (1990), and for Japanese by Germann (1999).

Although dependency grammars have been thoroughly investigated and successfully implemented in many different frameworks, phrase-structure grammars still dominate the field. It is instructive therefore to research the interaction of the two theories and, particularly, the influence of the dependency theory on the more a widely-used competing theory of constituency.

Various researchers have pointed out a shift to more dependency-like formalisms in the phrase-structure community. Hudson (1993) discusses three dominating linguistic theories of the 80s, Government-Binding theory (GB – Chomsky (1981)), Generalized Phrase-Structure Grammar (GPSG – Gazdar et al. (1985)) and Lexical-Functional Grammar (LFG – Bresnan (1982)). He shows that these phrase-structure theories demonstrate explicit trends to dependency in their syntax which involve:

- recognition of the notion of a *head* together with the principle stating that features of the head are projected up onto the mother phrase (GB, GPSG),
- recognition of relational categories, such as *subject* (LFG),
- recognition of dependency-like relations (*government* in GB).

More recently, a significant interest in dependency representations has been shown in statistical parsing. Thus, Collins (1996) describes a statistical parser based on probabilities of dependencies between head-words in the parse tree. Collins (1999) and Charniak (2000) use dependency relations for statistical disambiguation in constituency-based parsing.

Another important problem in computational linguistics in which dependency analysis have proven to be highly appropriate, namely evaluation of broad-coverage parsers, has been addressed by Lin (1995). The author has demonstrated the deficiency of evaluation methods based on phrase boundaries and proposed dependency-based evaluation which avoids problems common to phrase boundary evaluation and additionally has several desirable properties missing from other evaluation schemes. These properties include:

- ignoring the inconsequential differences between parser-generated parse trees and manually constructed parse trees,
- selective evaluation of given types of syntactic phenomena,
- a simpler diagnosis of incorrect parses.

In the proposed dependency-based evaluation, dependency relations are used in comparison of a parser output to a gold standard. The evaluation method is not restricted to dependency grammars: Lin (1995) additionally presents an algorithm for transforming constituency structures into dependency trees.

3.5 German dependency parsing

Three German dependency parsing models are described below: Topological Dependency Grammar developed in Saarland University, Weighted Constraint Dependency Grammar developed in Hamburg University and Concurrent Lexicalized Dependency Parser designed and implemented in Freiburg University.

3.5.1 Topological Dependency Grammar

Topological Dependency Grammar presented by Duchier and Debusmann (2001) is a constraint-based framework for dependency parsing of German. Dependency parsing is regarded in the framework as a constraint satisfaction problem with valid parse analyses being solutions of the problem. Initially, an input string is assigned all possible dependency parse analyses. A dependency analysis represents a dependency tree, nodes of which correspond to tokens of the string. The nodes are connected by labeled directed edges which indicate dependencies between tokens. The grammar formulates constraints on the grammaticality of parse analyses and outputs all parse analyses that satisfy the stated constraints.

The grammar represents a modular system with *immediate dependency* (ID) and *linear precedence* (LP) components. The immediate dependency component provides an input string with a *syntactic dependency tree* (ID tree) which is syntactically well-formed but which ignores issues of word order. The component is described in detail by Duchier (1999). The linear precedence component applies to a syntactic dependency tree and outputs an ordered, projective *topological dependency tree* (LP tree) which characterizes linear order of tokens in the input string.

Formally, Duchier (1999) defines a dependency grammar G as a 7-tuple:

$$< Words, Cats, Agrs, Comps, Mods, Rules, Lexicon >$$

where

- $Words$ is a finite set of strings representing word forms,
- $Cats$ is a finite set of POS categories,
- $Args$ is a finite set of agreement tuples such as $<$ *masc sing nom* $>$,
- $Comps$ is a finite set of complement role types such as `subject`,
- $Mods$ is a finite set of modifier role types, such as `adj` for adjectives,
- $Lexicon$ is a finite set of lexical entries, and
- $Rules$ is a family of binary predicates, indexed by role labels, expressing local grammatical principles.

A union of $Comps$ and $Mods$ sets represents a set of all role types $Roles$ which are used in the grammar for labeling dependencies.

A lexical entry is represented by an attribute value matrix which contains information about a token, its category, arguments and roles.

ID and LP trees consist of nodes and labeled edges. In an ID tree, edges are labeled with syntactic roles, such as *subject* or *vinf* (bare infinitival argument). In an LP tree, the edge labels represent topological fields markers, such as *mf* (middle field) or *vc* (verbal complex). Additionally, each node is assigned a lexical entry.

Figures 3.5 and 3.6 provide examples of an ID and LP trees for sentence (9), correspondingly. Since ID trees are unordered, an arbitrary linear arrangement is picked for expository purposes.

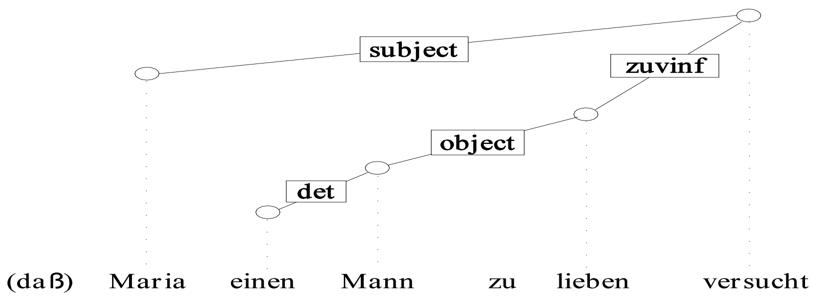


Figure 3.5: An ID tree in Topological Dependency Grammar

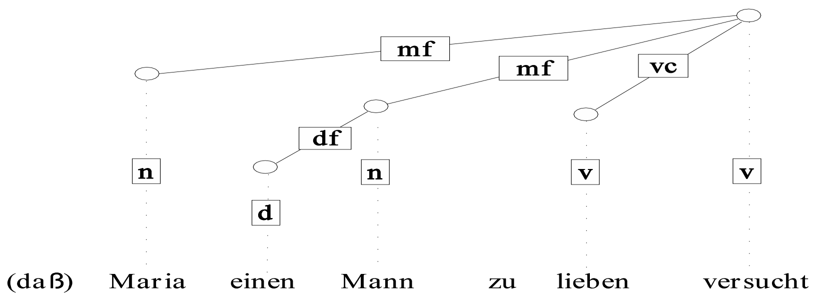


Figure 3.6: An LP tree in Topological Dependency Grammar

(9)	(daß)	Maria	einen	Mann	zu	lieben	versucht.
	(that)	Maria	a	man	to	love	tries

'(that) Maria tries to love a man.'

Various kinds of constraints, such as lexical constraints, valency constraints, role constraints, constraints on the form of the tree, node yield constraints and word-order constrains are stipulated by the grammar. The constraints restrict the form of the trees.

The parser and a German grammar fragment have been implemented using the constraint programming language Oz. Both parsing and generating modes are available.

3.5.2 Weighted Constraint Dependency Grammar

Similar to Topological Dependency Grammar, Weighted Constraint Dependency Grammar developed by Schröder et al. (2000) is based on constraint satisfaction techniques. However, it expands the formalism to a new dimension. While an ordinary constraint dependency grammar (CDG) does not allow the violation of constraints, a framework proposed by Schröder et al. (2000) introduces weakening, or softening, of constraints. Therefore, Weighted Constraint Dependency Grammar regards parsing as a constraint optimization problem rather than a constraint satisfaction problem. Using soft constraints provides valuable advantages over strict conditioning. First, a CDG with soft constraints allows for better modeling of performance aspects of natural language, since it encodes preferences rather than strict conditions and, therefore, is more robust. Soft constraints also provide an advantage of ranking parses and incorporating preferences into the analysis.

Weighted Constraint Dependency Grammar augments a pure CDG with a weighting function over the set of constraints. Thus, constraints are made dynamic in a sense that they are assigned different weights depending on the context. A second major difference between Topological Dependency Grammar and Weighted Constraint Dependency Grammar concern number of levels present in the grammars. While Topological Dependency Grammar deals exclusively with syntax, Weighted Constraint Dependency Grammar additionally provides a specification of functor-argument structure, where deep roles such as *agent, patient, theme* are annotated.

Formally, Schröder (2002) defines a weighted constraint dependency grammar as a 5-tuple:

$$< LEV, LAB, \alpha, C, \phi >$$

where

- LEV is a finite set of level symbols,
- LAB is a finite set of label symbols,
- α is a function that assigns a subset of label symbols to each level symbol,
- C is a set of constraints, and
- ϕ is a function that assigns weights to each constraint.

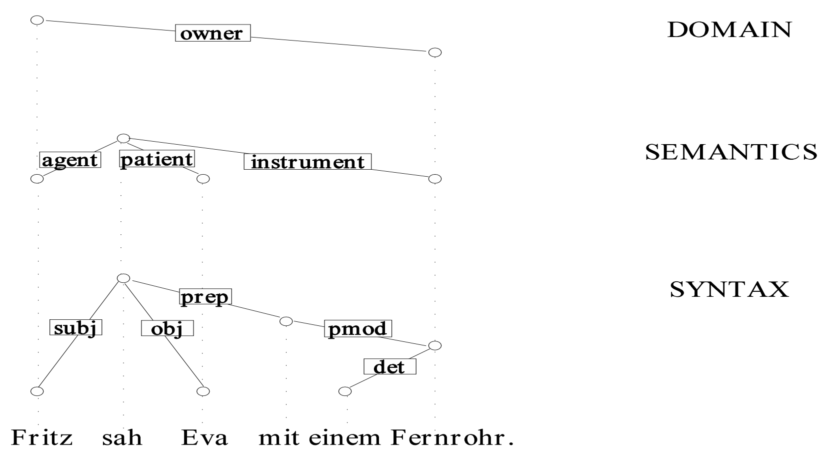


Figure 3.7: Structure provided by Weighted Constraint Dependency Grammar

A lexicon which associates a word form with additional morpho-syntactic, syntactic and semantic information is defined as a quadruple:

$$< W, \lambda, A, V >$$

where

- W is a finite set of word symbols,
- A is a finite set of attribute symbols,
- V is a finite set of value symbols, and
- λ is a function that assigns a attribute-value matrices to words.

Example structures of sentence (10) provided by Weighted Constraint Dependency Grammar are presented in Figure 3.7.

(10)	Fritz	sah	Eva	mit	einem	Fernrohr.
	'Fritz	saw	Eva	with	a	telescope'.

3.5.3 Concurrent Lexicalized Dependency Parser

A Concurrent Lexicalized Dependency Parser has been developed and described by Bröker et al. (1994), Schacht et al. (1994) and Hahn et al. (2000). It represents a radically lexicalized object-oriented model of natural language parsing.

The main principle of the model is the organization of all grammatical knowledge in lexical items which are treated as active *lexical processes* communicating with each other and dynamically establishing dependency relations between each other.

A lexical item contains information about the lexical and morpho-syntactic features of a token, its conceptual representation, valency constraints and, after successful parsing events, about governed lexical items and other grammatical relations, such as adjacency. Valency constraints stipulate the values of categorial, morpho-syntactic, conceptual and ordering features of the possible modifiers of the token. The lexicon of the model is organized as a lexical hierarchy where lexical items are represented as leaves and the intermediate nodes stand for generalized word classes.

The model is based on an object-oriented paradigm that assumes a collection of independent objects, the *actors*, which communicate via asynchronous pairwise message passing. Adapting the paradigm to the task of natural language processing, the actors are represented by lexical units. Dependency relations are established dynamically based on constraints formulated in the lexical units. A unit is searching for its head by checking for valency constraints of the other units. If the active unit satisfies the constraints of an addressed lexical unit, a head-modifier relation is built between them and the grammatical information about the units is updated correspondingly. In case of ambiguity, the structure is duplicated.

The described procedure results in incremental generation of dependency parse analyses for an input string. Similar to the parsers described above, Concurrent Lexicalized Dependency Parser is non-deterministic.

3.6 Conclusion

In this chapter, the theory of dependency has been presented. After introduction of a classical model of the theory, various issues in current dependency analysis have been discussed. The chapter has touched base on the computational aspects of the theory and provided a survey of German dependency parsing models.

Chapter 4

Xerox Incremental Parsing System

The Xerox Incremental Deep Parsing System (XIP) is a rule-based framework for robust syntactic analysis. The system has been developed at the Xerox Research Centre Europe in Grenoble and has been successfully applied for implementation of grammars for different languages. In the present thesis, the system is used for the creation of the German Incremental Parser (GRIP) described in Chapter 7 and for development of a rule-based morpho-syntactic tagger presented in Chapter 6.

The current Chapter represents a gentle introduction to the system. The purpose of the Chapter is to provide the reader with enough background for easier understanding of the structure and the potential of the morpho-syntactic tagger and of the GRIP parser described in the thesis. A more detailed description of the system is provided by Aït-Mokhtar et al. (2002).

Below, a general introduction to the XIP system and a description of the architecture of the system are given in Section 4.1. Section 4.2 describes the data representation in XIP. The kinds of rules used in the system are discussed in detail in Section 4.3. Section 4.4 concludes the Chapter.

4.1 Overview of the system

The XIP system is designed for the syntactic analysis of unrestricted text and can be used to provide analysis on any level of syntactic annotation ranging from morpho-syntactic annotation through chunking to deep parsing. The scope of analysis is not restricted to elements of a sentence and can be extended to establishing relations between elements of different sentences (such as pronouns and their antecedents) or between sentences themselves.

The underlying principle of the system is incrementality. The developers of the system Aït-Mokhtar et al. (2002) have argued that incrementality is a key to the robustness of syntactic analysis. They point out two major properties of incremental parsing systems which assure robustness and broad coverage: *self-containment* and *descriptive decomposition*.

The self-containment of rules in an incremental parsing system allows to avoid such side-effects of parsing as combinatory explosion, spurious ambiguity and parse failure. Descriptive decomposition provides means to tackle complex phenomena: higher levels of annotation rely on the output of the lower levels and are, therefore, less sensitive to the complexity and variability of the input string. Thus, the system preserves the robustness of shallow parsers and provides a deeper level of linguistic analysis.

The incrementality of the XIP formalism is ensured by the modular architecture of the system and by the layered organization of rules inside each module. An input string passes through rule layers and with each rule application, the analysis of the string is refined. In this regard, the XIP system is similar to cascaded finite-state parsers.

The XIP formalism offers an advantage of using feature lists, which provide a means for specification of fine-grained regularities in the rules.

4.1.1 XIP architecture

The XIP system consists of two optional pre-processing modules and four proper XIP modules.

The first pre-processing module is a Normalization, Tokenization and Morphology (NTM) module. The module provides the normalized form and all potential lexical information for each token. The morphological component of the module is represented by the Xerox Morphological Analyzer.

The second pre-processing module is a Hidden Markov Model (HMM) disambiguator. The disambiguator uses the HMM algorithm for providing the most likely analysis for each token based on the context. The standard package for German includes the HMM module. However, in the GRIP system described in the thesis, the module is substituted by a rule-based disambiguator, since it proves to provide a better performance.

The first proper XIP module is an input control module which pretreats the input text before it is processed by other modules. Among other tasks, the module defines a processing unit of the system (clause, sentence, paragraph, etc.) and determines the mechanism of feature percolation.

The following disambiguation module is based on disambiguation rules

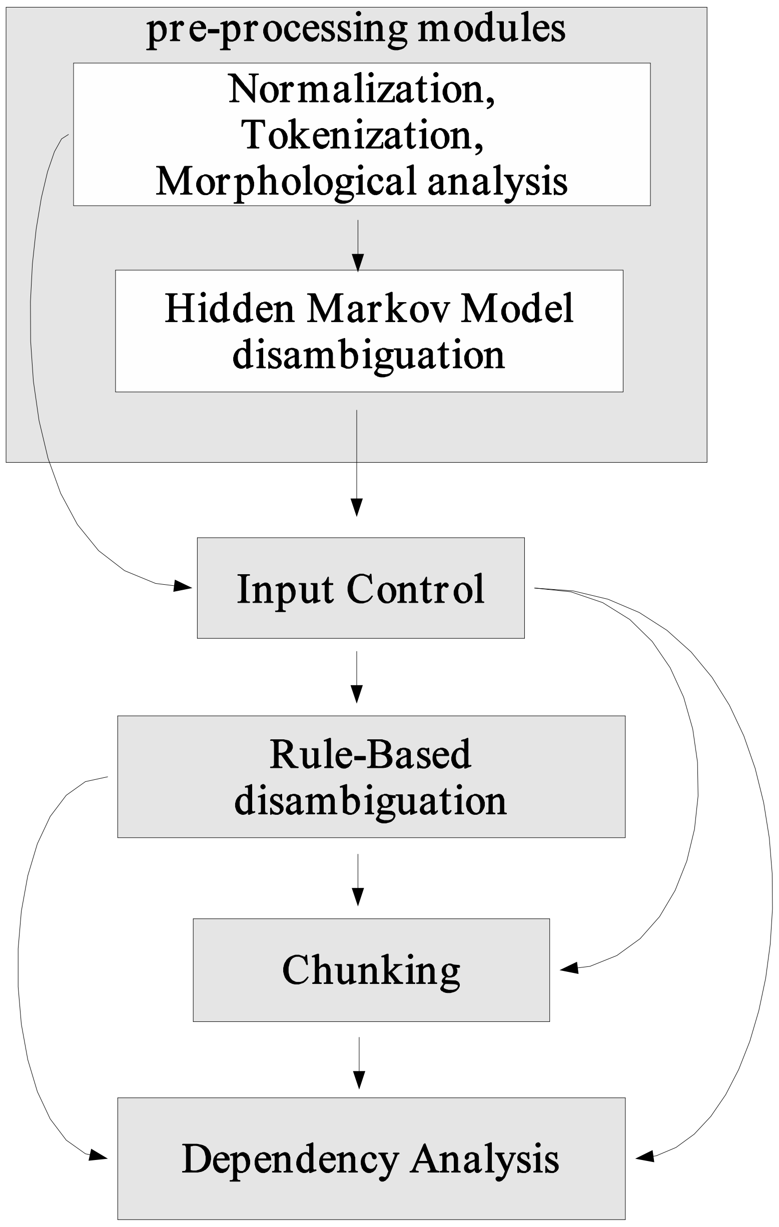

pre-processing modules
Normalization, Tokenization, Morphological analysis
Hidden Markov Model disambiguation
Input Control
Rule-Based disambiguation
Chunking
Dependency Analysis

Figure 4.1: Architecture of the XIP system

which select valid readings for tokens given the context. The module can be used in conjunction with the HMM module or separately.

The chunking module applies next. Rules of the chunking module create phrase structures over the input string.

The final dependency module produces dependency relations between tokens. Such relations include, but are not restricted to, function argument dependencies, such as `subject` and `object`. Further relations that can be built into the module may extend beyond sentence boundaries.

The system can be used for the creation of separate syntactic analyzers: taggers, chunkers, constituent parsers and dependency parsers. Any number of XIP modules can be combined in the system. Thus, a dependency module can be built directly on the output of the rule-based morphological disambiguator, without additional pre-structuring of the input into syntactic trees.

Moreover, the system allows for any kind of input: raw input, morphologically analyzed input, morphologically disambiguated input and even syntactically annotated text. A XIP grammar maps categories of the input into its own categories and features. Thus, no pre-processing of the data is needed. External components, such as a tagger or a chunker, can also be plugged into the XIP architecture.

Additionally, a XIP grammar may contain a lexicon with category- and feature-value pairs for tokens, which provides a means for adding or redefining analyses supplied in the input.

A general scheme of the XIP architecture is presented in Figure 4.1.

4.2 XIP data representation

An elementary unit in data representation of the XIP system is a node. It is defined by:

- a category,
- feature-value pairs,
- pointers to sister nodes,
- pointers to daughter nodes (in case of constituent nodes).

Categories and features should be pre-defined by a grammar developer.

4.2.1 Operations on nodes

Nodes can be combined into sequences, compared to each other and explored as to their inner structure. Table 4.1 provides a list of possible operations that define node sequences.

Operator	Description
,	Concatenation
()	Optionality
*	Kleene star
+	Kleene plus
?	Any category
;	Disjunction
∼	Negation

Table 4.1: Operators for defining node sequences in XIP

All operators except for negation can be combined. An example sequence of nodes "`det, ?*, noun;card`" stands for all sequences that start from a determiner and end with a noun or a cardinal. The disjunction operator functions *in situ*: i.e. a template "`det;adj, noun`" means "det OR adj, noun". The negation operator specifies that a category cannot exist in a particular context. Thus, "∼`noun`" means any category except for a noun.

Operators for comparing nodes are presented in Table 4.2.

Operator	Description
::	equivalence
∼:	difference
<	precedence
>	following

Table 4.2: Operators for comparing nodes in XIP

The inner structure of a tree can be explored with the {} operation. Thus, an expression `NP{det,?*,noun}` describes a noun phrase, the left-most daughter of which is a determiner and the right-most daughter of which is a noun. The node can have other daughters which are placed between a determiner and a noun.

Another useful characteristic of the system is the possibility to associate nodes with variables. Use of variables provides a necessary basis for comparing nodes and testing their feature values.

4.2.2 Operations on features and categories

The XIP system allows for instantiation, creation, deletion and comparison of features and categories. Additionally, values of features and categories can be tested. Advanced operations on features include, among others, modifying features and percolating features to mother nodes.

Table 4.3 summarizes possible basic operations on features and categories.

Operator	Description
[feature:value]	presence of the feature and the value
[feature:∼value]	absence of the value on the feature
[feature]	the feature is instantiated
[feature:∼]	the feature is not instantiated
|	Boolean OR
&	Boolean AND

Table 4.3: Operators on features and categories in XIP

Table 4.4 describes operations on feature values allowed in XIP.

Operator	Description
:	features have a common subset of values
::	sets of values of two features are identical
∼:	features have no common subset of values
$<$, $>$, $\leq$, $\geq$	comparison of feature values, operators also specify a domain over which the value for a given feature should be valid

Table 4.4: Operators on features values in XIP

4.3 Different types of rules in XIP

In the XIP system, different types of rules are available. Disambiguation rules serve for restricting sets of possible analyses of lexical nodes. Chunking rules combine nodes in phrase structures. With reshuffling rules, phrase-structure subtrees can be re-organized. Marking rules can be used to mark specific node configurations with feature-value pairs. Dependency rules establish relations between nodes.

The general format of rules includes:

- a filter,
- (optional) context fields,
- (optional) conditions,
- specification of a rule action.

A filter determines a set of units to which the rule applies. For example, in case of disambiguation rules, a filter can represent a set of possible readings of a lexical node. A rule will apply to all nodes that have readings specified in its filter. In case of chunking rules, a filter is defined as a list or a set of nodes. All sequences of nodes specified in the filter will be considered and if contextual constraints and conditions on feature values are satisfied, the nodes will be combined under a common root. A more detailed description of a filter field, as well as other fields, specifically for each kind of rules is given in corresponding subsections below.

A contextual environment of units to which a rule applies can be specified in the right and left context fields. The context fields are delimited by pipe signs (||) and are defined as sequences of nodes. Specification of an empty context, as well as negation of a context, is also possible.

Further constraints on features across different nodes in a rule can be specified in the condition field. This is made possible by the use of variables in the rule. An example of the condition which requires identity of *agreement* values on a determiner and a noun is shown in (11):

(11) det#1, adj*, noun#2, where (#1[agreement] :: #2[agreement]).

A rule applies only if its contextual constraints and constraints stipulated in the condition field are satisfied. Formal specification of the rule action depends on the type of rule and is described for each rule type independently in corresponding subsections below.

4.3.1 Disambiguation rules

Two types of rules can be used for disambiguation in the XIP system: ordinary disambiguation rules and double reduction rules.

Ordinary disambiguation rules (ODRs) apply to lexical nodes with multiple readings and select readings that are valid in a given context. The general format of ODRs is presented in (12):

(12) readings_filter = |left_context| selected_readings |right_context|.

The readings filter determines the domain of application of an ODR. It is defined by a set of categories with possible specification of feature values. Left and right context fields restrict the context of rule application. As the name suggests, the set of readings selected by the rule is specified in the field `selected_readings`.

The second type of disambiguation rules, namely double reduction rules, have been designed specifically for the development of the morpho-syntactic tagger for German in the XIP system. The German language is notorious for its morphological ambiguity. Consider an example phrase "*Die schönen Blumen*" ("*the beautiful flowers*") and its analyses in (13):

(13)	Die	+Det+Art+Pl+FMN+Acc+St+ART
	Die	+Det+Art+Pl+FMN+Nom+St+ART
	Die	+Det+Art+Sg+Fem+Acc+St+ART
	Die	+Det+Art+Sg+Fem+Nom+St+ART
	schönen	+Adj+Pos+Pl+FMN+NGDA+Wk+ADJA
	schönen	+Adj+Pos+Pl+FMN+Dat+St+ADJA
	schönen	+Adj+Pos+Sg+FMN+Dat+Wk+ADJA
	schönen	+Adj+Pos+Sg+FMN+Gen+Wk+ADJA
	schönen	+Adj+Pos+Sg+Masc+Acc+St+ADJA
	schönen	+Adj+Pos+Sg+Masc+Acc+Wk+ADJA
	schönen	+Adj+Pos+Sg+Masc+Gen+St+ADJA
	schönen	+Adj+Pos+Sg+Neut+Gen+St+ADJA
	Blumen	+Noun+Common+Pl+Fem+NGDA+NOUN

The disambiguation of lexical nodes in such a phrase with ordinary disambiguation rules is possible but would require a statement of multiple rules for the comparison of readings on the nodes. An example of a rule of this kind is given in (14). It disallows singular number readings on a noun if the preceding adjective is not singular. Similar rules for other features and feature values have to be present in the grammar for successful disambiguation, as well as similar rules with instantiations of different categories, such as a determiner instead of an adjective, for example. Altogether, 18 rules need to be included in the grammar to ensure the strict identity of agreement for the feature values of the three nodes.

(14) noun = |adj[sg:∼]| noun[sg:∼].

Double reduction rules provide a simple mechanism for eliminating non-shared readings among categories. For computational efficiency, features on only two categories can be compared at a time. The general format of a double reduction rule is presented in (15):

(15) |node_sequence| ⇒ boolean_constraints.

The `node_sequence` field serves to specify the context in which the rule applies, and with which lexical nodes they are to be considered. These nodes are represented as part of the context and are associated with variables. The `boolean_constraints` field contains constraints on the feature values of the nodes considered.

An example rule (17) instantiates the general format of double reduction rules to the disambiguation of lexical nodes inside a phrase, such as in (13):

(16) |adj#1,noun#2| ⇒ (#1[agr] : : #2[agr]).

The rule checks agreement values (such as *case, number* and *gender*) on a noun and a preceding adjective and selects only those readings which bear the same values. Application of the rule leads to disambiguation of two lexical nodes simultaneously.

Joint application of the rule together with two other double reduction rules which compare readings on a determiner and a following adjective and on a noun and a preceding determiner will guarantee that only shared readings are kept on a determiner, an adjective and a noun following each other. The context of nodes in the `node_sequence` field of a double reduction rule can be expanded so that it allows for an intermediate material:

(17) |det#1,(adv),adj*,noun#2| ⇒ (#1[agr] : : #2[agr]).

If no shared readings are present on the nodes compared in the rule, the rule does not change the sets of analyses of the nodes. This strategy ensures that no node is deprived of all its readings.

Disambiguation rules are organized in layers. A layer can contain an unrestricted number of either ODRs or double reduction rules, but not rules of both type at the same time. Rules with the most specific filter apply first. If several rules have the same filter, the order of rule application is determined by the order of rules in the grammar file. Once the rules of a layer have applied, the input string is passed to the next layer. To insure the serial application of a rule, it should be stated repetitively in different layers. This property distinguishes XIP system from context free grammars, in which the presence of a rule in a grammar enables multiple application of the rule.

4.3.2 Chunking rules

Chunking rules can be used for grouping nodes into trees. No restriction on depth or tree complexity is imposed by the XIP formalism, so that complete phrase-structure trees can be created in the chunking module.

Similar to disambiguation rules, chunking rules are organized in layers. The rules are of a deterministic nature: once a structure has been assigned, it is never dismissed. The resulting structure is passed to the next layer. Embedded and recursive structures can be created by the repetitive statement of rules in different layers. Thus, for the correct analysis of a noun phrase with embedded prepositional phrase, such as "*ein auf Südamerika spezialisiertes Reiseunternehmen*" presented in (18), the following rules are required:

- NP -> (det), AP*, noun.
- PP -> prep, NP.
- AP -> PP, adj.
- NP -> (det), AP*, noun.

(18)	ein	auf	Südamerika	spezialisiertes	Reiseunternehmen
	a	on	South America	specializing	travel company

‘a travel company specializing on South America’

Three types of chunking rules are present in the system: immediate dominance (ID) rules, linear precedence (LP) rules and sequence rules. ID rules define a set of nodes to be combined under the same root. LP rules cooperate with ID rules to determine a linear order of nodes to be combined. ID and LP rules are the only types of rules that can and should be defined on the same layer.

ID rules have the following format:

(19) new_node -> |left_context| list_of_lexical_nodes |right_context|.

Here, the `new_node` field determines a mother node which will be created to combine the lexical nodes specified in the `list_of_lexical_nodes` field.

LP rules impose constraints of the order of lexical nodes in the corresponding rule layer. The format of the LP rules is the following:

(20) [set of features] < [set of features]

The rule defines that nodes with a set of features in the left-hand side of the rule should precede nodes with a set of features in the right-hand side of the rule. Thus, rule (21) requires that determiners precede nouns in ID rules:

(21) [det:+] < [noun:+]

Another possibility to restrict the order of lexical nodes in an ID rule is by using the features `[first]` and `[last]` on those lexical nodes that begin and end a node sequence to be combined under a common mother node. Rule (22) exemplifies the use of these features:

(22) NP -> det[first], AP*, noun[last].

In a given layer, the first rules that apply define the longest sequence of nodes. If several rules compete for the longest match, the order of rules in the layer specifies the order of rule application.

The third type of chunking rules, sequence rules, perform in the same manner as ID rules: they group a sequence of nodes together. The difference between the two rule types lies in the fact that sequence rules define a list of lexical nodes to be combined, i.e. sequence rules determine an order of nodes. Sequence rules are also flexible as to the direction of processing of an input string (left to right or right to left) and as to the choice between the longest and the shortest match.

The general format of sequence rules is presented in (23):

(23) new_node = |left_context| list_of_lexical_nodes |right_context|.

The equivalence sign `=` can be substituted for another operation. The semantics of different operations possible in sequence rules is defined in Table 4.5:

4.3.3 Marking and reshuffling rules

Marking and reshuffling rules are advanced rules for modifying structures created by chunking rules. Marking rules add features to a node based on

	direction of processing an input string	
	left to right	right to left
shortest match	=	<=
longest match	@=	@<=

Table 4.5: Operations available for XIP sequence rules.

the node's inner structure. Thus, a subtree can be marked with feature *passive*, if it consists of the auxiliary verb "*be*" and a past participle.

With reshuffling rules, a chunk structure can be reorganized: a subtree is re-assigned to a different mother node. The use of reshuffling rules can be illustrated on the example of an analysis of a noun phrase with an embedded prepositional phrase, presented above in (18). Grammar rules relevant for the analysis are:

- AP -> (adv), adj[last].
- NP -> (det), AP*, noun[last].
- PP -> prep, NP[last].
- AP -> PP, adj[last].
- NP -> (det), AP*, noun[last].

A very careful and intricate specification of context in the rules is required so that a correct phrase structure presented in (24) can be build:

(24) NP{ein AP{PP{auf NP{Südamerika} spezialisiertes}} Reiseunternehmen}

If the rules given above are applied without any context specification or if the context constraints are not elaborate enough, the following structure will be produced:

(25) ein PP{auf NP{Südamerika}} NP{AP{spezialisiertes} Reiseunternehmen}

A simple reshuffling rule can be stated that re-assigns the prepositional phrase under the AP node and includes the determiner into the second NP, resulting in the correct structure. Statement of such a rule allows for avoidance of the definition of complicated context constraints in the chunking rules and for the elimination of the second rule for adjectival phrase (AP).

4.3.4 Dependency rules

Dependency rules are designed for establishing relations between nodes. They also can establish unary relations, assign features to nodes and delete or rename a previously defined relation.

The general format of dependency rules is the following:

(26) |pattern| if <conditions> <dependency_terms>.

The `pattern` field combines a filter and a context fields. It specifies a node sequence and associates one or more nodes with variables. Reference to the feature values of nodes and exploration of the inner structure of nodes is possible.

The `dependency_terms` field defines a new dependency to be created. It consists of a name for dependency relation and an n-ary set of variables.

The `conditions` field is an optional field that represents a Boolean expression over dependencies. In this field, the existence of other dependency relations and their inter-connections can be checked. If a rule is used for modifying or deleting a previously defined dependency relation, the relation to be modified or to be deleted is marked in the `conditions` field and the `dependency_terms` field determines whether the relation is to be deleted (with a $\sim$ sign) or to be renamed (with a new dependency term).

4.4 Conclusion

The current Chapter has introduced the Xerox Incremental Deep Parsing System, which is used for the development of a morpho-syntactic tagger and a dependency parser in the thesis. Section 4.1 has described the underlying principles and the general architecture of the system. Data representation of the system and the types of rules available in the system have been discussed in Section 4.2 and Section 4.3, correspondingly.

Chapter 5

The TüBa-D/Z treebank

The TüBa-D/Z treebank (Telljohann et al. (2003)) has supplied data for all of the experiments with statistical taggers described in this thesis. Additionally, the annotation scheme adopted for the German dependency parser presented in this thesis is based on the TüBa-D/Z annotation scheme. Finally, TüBa-D/Z data have been used for the evaluation of the taggers and the dependency parser described in this thesis. The current chapter aims at familiarizing the reader with the treebank.

General information, including a brief summary of principles which have guided the treebank annotation and a description of the structure of the treebank, is introduced in Section 5.1. Sections 5.2 – 5.4 describe different annotation levels of the treebank: morpho-syntactic information, constituency structure and dependency relations between constituents. Section 5.5 concludes the chapter.

5.1 General information on the treebank

The name of the treebank, TüBa-D/Z, represents an abbreviation of the full name, *die Tübinger Baumbank des Deutschen / Schriftsprache* (the Tübingen Treebank of Written German). The treebank material consists of data taken from the German newspaper 'die tageszeitung' (taz (1999)), editions from May 3rd to May 7th 1999.

Together with the Tübingen Treebank of Spoken German (TüBa-D/S), formerly called the Verbmobil German Treebank, TüBa-D/Z constitutes the Tübingen Treebank of German.

The annotation principles of Verbmobil conform to the annotation scheme adopted in the Verbmobil project (Hinrichs et al. (2000), Stegmann et al.

(2000)). Additionally, systematic extensions of the Verbmobil scheme have been introduced to accommodate annotation of TüBa-D/Z to the characteristics of written texts.

In the inventory of categories design, the TüBa-D/Z principles have been based on considerations of linguistic adequacy, of theory-neutrality and on processing considerations. In line with these considerations, the following decisions about the annotation scheme have been made:

- empty categories and crossing branches are avoided in the treebank;
- sets of node and edge labels used in the treebank reflect empirical generalizations identified by syntacticians as characteristic of German;
- the annotation is based on the notion of *topological fields*;
- constituents are grouped together based on the following three common principles:
  - the *flat clustering principle*,
  - the *longest match principle*,
  - the *high attachment principle.*

Absence of empty categories and of crossing branches in the treebank ensures the usability of the treebank data for parsers that rely on context-freeness of the underlying grammar.

The topological fields framework adopted in the treebank has a rich tradition in descriptive studies of German syntax (Herling (1821), Erdmann (1886), Drach (1937), Höhle (1985)). It aims at capturing the fundamental word-order regularities of German sentence structure and provides a theory-neutral inventory for description of word regularities.

The *flat clustering principle* guarantees the minimal number of hierarchy levels possible in the syntactic structure. The *longest match principle* ensures the combination of a maximally possible number of daughters under the same mother node. According to the *high attachment principle*, ambiguous modifiers are attached to the highest possible level in a tree structure. In the annotation process, satisfaction of the principles have been imposed unless resulting constructions are syntactically or semantically ill-formed.

In the treebank, grammatical information is annotated on three levels: *morpho-syntactic level, constituency level* and *dependency level.* The morpho-syntactic level represents the annotation of lexical nodes with POS information and inflectional morphology. The constituency level comprises

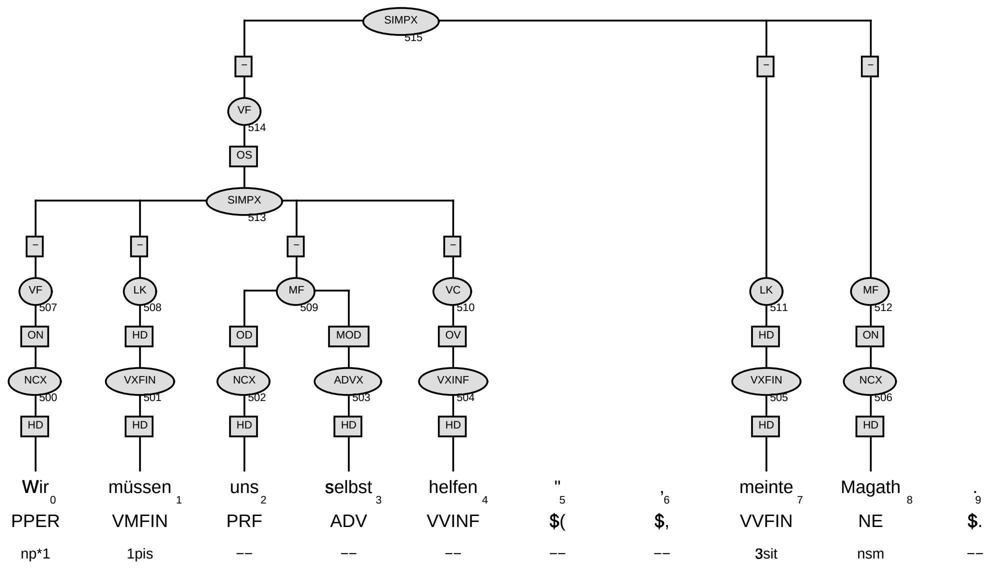


Figure 5.1: An example tree from the TüBa-D/Z treebank

phrase structure information. Grammatical functions of individual phrases and syntactic dependencies between phrases are annotated on the dependency level.

An example tree from the TüBa-D/Z treebank is demonstrated in Figure 5.1[1]. The tree provides a syntactic analysis for sentence (27). Relevant morpho-syntactic information is presented under the lexical nodes. The syntactic tree consists of a set of labeled nodes and a set of labeled edges which connect nodes. Node labels are enclosed in ovals and are (optionally) marked with node numbers. Edge labels encode dependency relations and are given in rectangular forms. A detailed description of the TüBa-D/Z annotation levels is provided in corresponding sections below.

(27)	"Wir	müssen	uns	selbst	helfen",	meinte	Magath.
	"we	must	us	ourselves	help",	meant	Magath

'"We must help ourselves", said Magath.'

As the example tree demonstrates, neither sentence-final nor sentence-internal punctuation is annotated in TüBa-D/Z. The only exception to this

[1]The tree diagram, as well as all tree diagrams in the current chapter, has been generated with the aid of the Negra *Annotate* tool (Plaehn (1998)).

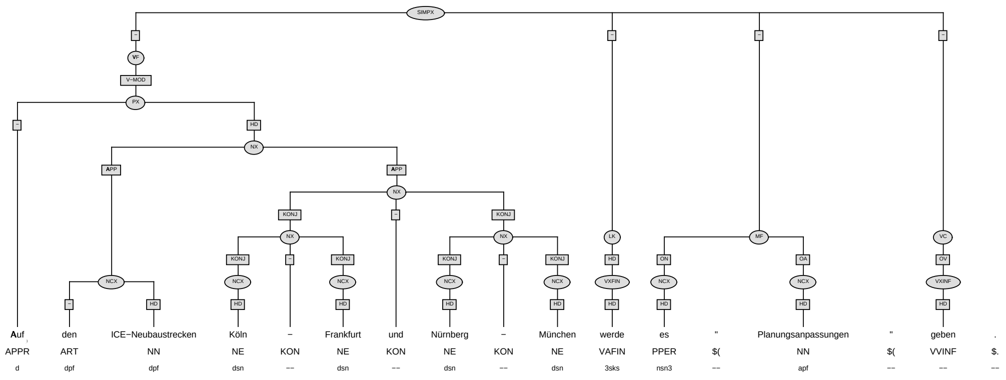

SIMPX
VF
V-MOD
PX
HD
NX
APP
NCX
KONJ
LK
VXFIN
MF
ON
OA
VC
OV
VXINF
Auf den ICE-Neubaustrecken Köln – Frankfurt und Nürnberg – München werde es " Planungsanpassungen " geben .
APPR ART NN NE KON NE KON NE KON NE VAFIN PPER $(NN $(VVINF $.
d dpf dpf dsn -- dsn -- dsn -- dsn 3sks nsn3 -- apf -- -- --

Figure 5.2: A TüBa-D/Z tree with annotated punctuation

decision represent punctuation with semantic meaning, such as, for example, a hyphen in "*15.30 – 17.30 Uhr*", which stands for "*bis*" ("*till*"), or in "*Köln – Frankfurt*" (consider an example tree in Figure 5.2 which corresponds to sentence (28)). Semantically marked punctuation also receives a different POS tag: APPR (preposition) or KON (coordinate conjunction) instead of `$(` (parenthetic punctuation).

(28) Auf den ICE-Neubaustrecken Köln – Frankfurt und Nürnberg –
on the newly built ICE lines Köln – Frankfurt and Nürnberg –
München werde es "Planungsanpassungen" geben.
München will it "plans adjustments" give
'The plans will be adjusted on the newly built ICE lines Köln – Frankfurt and Nürnberg – München.'

In line with the considerations of most syntactic theories, a segmentation unit of syntactic annotation in TüBa-D/Z is assumed to be a complete sentence, i.e. a syntactic unit delimited by punctuation marks {`.` `!` `?` `;` `-` `...` `/`}. However, due to a number of phenomena specific for newspaper texts, a segmentation unit may be extended to incomplete sentences, phrases and to combinations of sentences and/or phrases. Phenomena that lead to such extensions include headlines, titles, parentheses, discourse markers and sentence conjunction by a colon. An example segmentation unit which represents a headline is demonstrate in Figure 5.3.

(29) Die etwas andere Boulevardkomödie: Oliver Bukowskis derbes
The slightly other boulevard comedy: Oliver Bukowski's rough

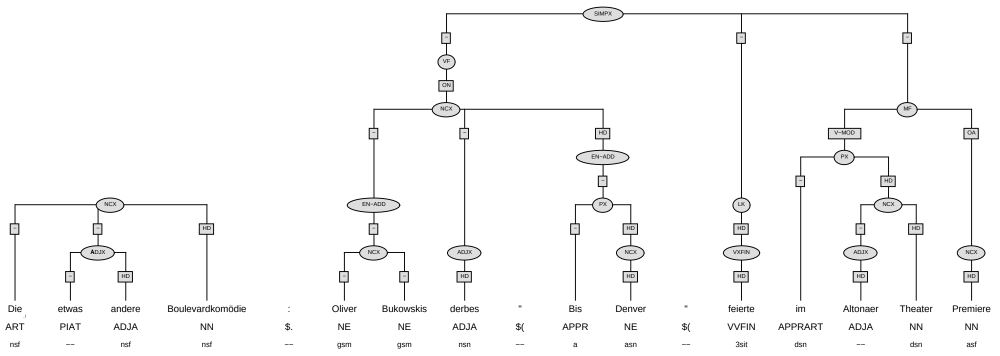

SIMPX
VF
ON
NCX
EN-ADD
ADJX
PX
LK
VXFIN
MF
V-MOD
OA
HD
Die etwas andere Boulevardkomödie : Oliver Bukowskis derbes " Bis Denver " feierte im Altonaer Theater Premiere
ART PIAT ADJA NN $. NE NE ADJA $(APPR NE $(VVFIN APPRART ADJA NN NN
nsf -- nsf nsf -- gsm gsm nsn -- a asn -- 3sit dsn -- dsn asf

Figure 5.3: A TüBa-D/Z segmentation unit representing a headline

“Bis Denver” feierte im Altonaer Theater Premiere
“Bis Denver” celebrated in the Altonaer Theater premiere
‘A slightly different boulevard comedy: Oliver Bukowski’s rough “Bis Denver” celebrated the premiere in the Altonaer Theater.’

Discourse markers, such as “*sagt einer*” (“*says somebody*”) in sentence (30), represent a separate tree in TüBa-D/Z analysis, even if they occur inside a complete sentence. A segmentation unit which contains a discourse marker is shown in Figure 5.4.

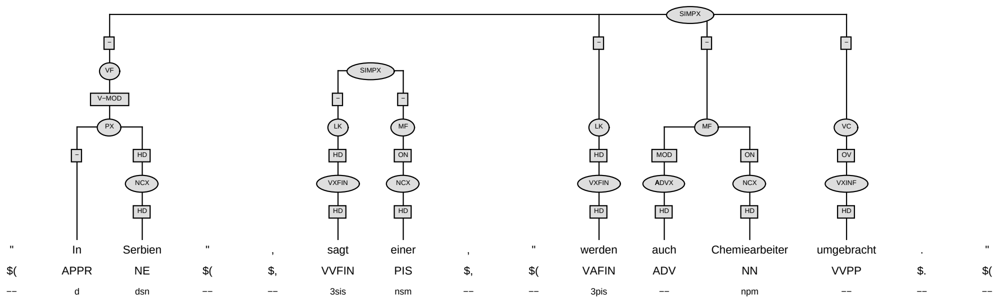

SIMPX
VF
V-MOD
PX
HD
NCX
LK
VXFIN
MF
ON
MOD
ADVX
VC
OV
VXINF
" In Serbien " , sagt einer , " werden auch Chemiearbeiter umgebracht . "
$(APPR NE $($, VVFIN PIS $, $(VAFIN ADV NN VVPP $. $(
-- d dsn -- -- 3sis nsm -- -- 3pis -- npm -- -- --

Figure 5.4: A TüBa-D/Z segmentation unit with a discourse marker

(30) “In Serbien”, sagt einer, “werden auch Chemiearbeiter
“in Serbia”, says somebody, “get also chemical workers
umgebracht.”
murdered”

'"In Serbia", says somebody, "chemical workers are also murdered".'

The TüBa-D/Z treebank currently consists of 15 260 segmentation units (approximately 270 000 tokens). Table 5.1 presents statistics over the length of segmentation units in TüBa-D/Z.

Minimum	Maximum	Mean	Mode
1	104	17.5	14

Table 5.1: Length of TüBa-D/Z segmentation units

5.2 Morpho-syntactic level of annotation

On the morpho-syntactic level, lexical nodes are annotated with POS information and with markers of inflectional morphology.

The POS tagset used in TüBa-D/Z is the Stuttgart-Tübingen Tagset (STTS, Schiller et al. (1995)), a widely accepted tagset for German. The tagset distinguishes 51 tag for lexical tokens and 3 punctuation tags. A full list of STTS tags together with their description is provided in Appendix A.

Morphological features employed in the treebank include *case, number, gender, person, mood* and *tense*. All lexical tokens which exhibit inflectional morphology are assigned a cluster of feature-value pairs. Thus, nouns, adjectives, determiners and non-personal pronouns are annotated with *case, number* and *gender* information. Finite verbs are provided with *person, number, mood* and *tense* information. A complete list which describes feature combinations for each STTS tag is presented in Appendix C.

The values of morphological features are presented in the treebank explicitly. Features that correspond to the values can be uniquely identified by the position of a value in a cluster, provided the POS tag. Thus, a cluster `3sis` assigned to verb "*sagt*" in Figure 5.4 stands for "**3**rd person, **s**ingular number, **i**ndicative mood, pre**s**ent tense".

In total, 433 distinct morphological value clusters can be generated. Together with POS value, they result in a tagset of 1 317 tags. The actual number of tags which occur in the treebank amounts to 555.

The process of morphological annotation of TüBa-D/Z was initiated at the beginning of the current dissertation project. At the completion of the project experiments, the number of annotated segmentation units in TüBa-D/Z amounted to 11 000, which corresponds to approximately 200 000 tokens.

5.3 Constituency level of annotation

The constituency level of annotation corresponds to syntactic analysis in terms of constituency structure. Three sublevels of constituency annotation are distinguished in TüBa-D/Z: *sentence sublevel*, *field sublevel* and *phrase sublevel*.

On the sentence sublevel, root nodes are marked as different types of clauses or as a discourse marker. The types of clauses distinguished in TüBa-D/Z are *simplex clause, relative clause* and a *paratactic construction of simplex clauses.*

Immediately below the sentence nodes, the nodes of topological fields are located. Depending on the position of a finite verb in a clause, three types of German clauses are distinguished: *verb-initial* (V-1), *verb-second* (V-2) and *verb-final* (V-end). While in verb-final clauses a finite verb constitutes a continuous unit with non-finite verbal elements, in verb-initial and verb-second clauses, a verbal complex represents a discontinuous structure (consider examples in Figures 5.1, 5.2 and 5.4). According to the theory of topological fields, a German clause is structured into fields. Verbal elements form sentence brackets and divide a clause into an initial field (`VF`), a middle field (`MF`) and a final field (`NF`). A finite verb represents a left sentence bracket (`LK`), whereas a verbal complex (`VC`) serves as a right sentence bracket. Additional fields are:

- `KOORD` for clause-initial coordinating particles,
- `PARORD` for clause-initial non-coordinating particles,
- `LV` for resumptive constructions,
- `C` for complementizers.

Topological schemes for the three types of German clauses are presented in Table 5.2. In practice, any of the fields can be omitted. Sentence (31) provides an example of a verb-initial clause. Examples of a verb-second and a verb-final clause are presented by sentences (32) and (33), correspondingly.

(31)	Habe	ich	nicht	schlecht	gedacht	über	dieses	und	jenes?
	'Have	I	not	badly	thought	about	this	and	that'
	`LK`	`MF`			`VC`	`NF`			

Haven't I thought badly about this and that?

V-1	`KOORD - LV - LK - MF - VC - NF`
V-2	`KOORD/PARORD - LV - VF - LK - MF - VC - NF`
V-end	`KOORD - C - MF - VC - NF`

Table 5.2: Topological schemes for German clauses

(32)

Das	muß	man	spielen	als	Deutscher.
'This	must	one	play	as	German'
`VF`	`LK`	`MF`	`VC`	`NF`	

One should play it as a German.

(33)

Jemand,	der	sein	Studium	ordentlich	abgeschlossen	hat.
Someone,	who	his	studies	properly	finished	has
	`C`	`MF`			`VC`	

'Someone who has finished his studies properly'.

On the phrase level, phrasal constituents are annotated. Maximal phrasal categories are identified by the character `X` in the node label, preceded by the abbreviation for the type of phrase. E.g. `PX` is the TüBa-D/Z treebank equivalent of the more traditional label `PP`. Two types of noun phrases are distinguished in TüBa-D/Z: `NCX` and `NX`. The former stands for a non-recursive noun phrase, whereas the latter represents a noun phrase of a higher level that includes an `NCX` node in it. An example of a subtree that contains both `NCX` and `NX` nodes is presented in Figure 5.5. The example additionally illustrates the use of the node label `EN-ADD` which is introduced in the treebank to mark proper nouns and named entities.

(34)

"Gleichstellung	statt	Barrieren"
"equal opportunities	instead of	barriers"

"'Equal opportunities instead of barriers".'

A complete list of TüBa-D/Z node labels is presented in Appendix D.1.

5.4 Dependency level of annotation

The syntactic relations between constituent nodes are annotated on the dependency level. The relations distinguished in TüBa-D/Z include verbal complements, such as *subject, direct object*, and modifiers, such as *verbal modifier, modifier of a dative object*. Additionally, constituents are identified

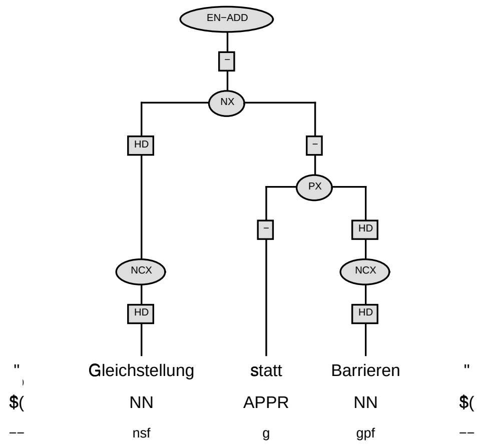


Figure 5.5: A TüBa-D/Z subtree for a named entity

with a head marker (`HD`) in case they perform a function of a head in a phrase, or with a non-head marker (`-`), otherwise. In coordination constructions, each conjunct depends on the head of the whole construction and is marked as `KONJ`. The head of a clause is a finite verb.

Syntactic relation information is encoded as labels on edges in syntactic trees. Long-distance dependencies are marked in the treebank by special edge labels that encode information both about the type of a relation and about the head constituent. Thus, to mark a modifier relation between a prepositional phrase and its head in sentence (35), a label `OA-MOD` (*modifier of the accusative object*) is used. The label refers to a accusative object constituent present in the same tree. By such a reference, the label unambiguously identifies the dependent-head relation. The TüBa-D/Z analysis for sentence (35) is given in Figure 5.6.

(35)

Widersprüchliche	Angaben	gab	es	über	ein	angebliches
contradictory	information	gave	it	about	a	alleged

Teilgeständnis	des	mittlerweile	entlassenen	Präfekten.
partial confession	the$_{gen}$	meanwhile	dismissed	prefects

‘There was a contradictory information about an alleged partial confession of the prefects dismissed in the meanwhile. ’

In addition, the treebank employs a set of secondary edge labels for ambiguity resolution in the following cases:

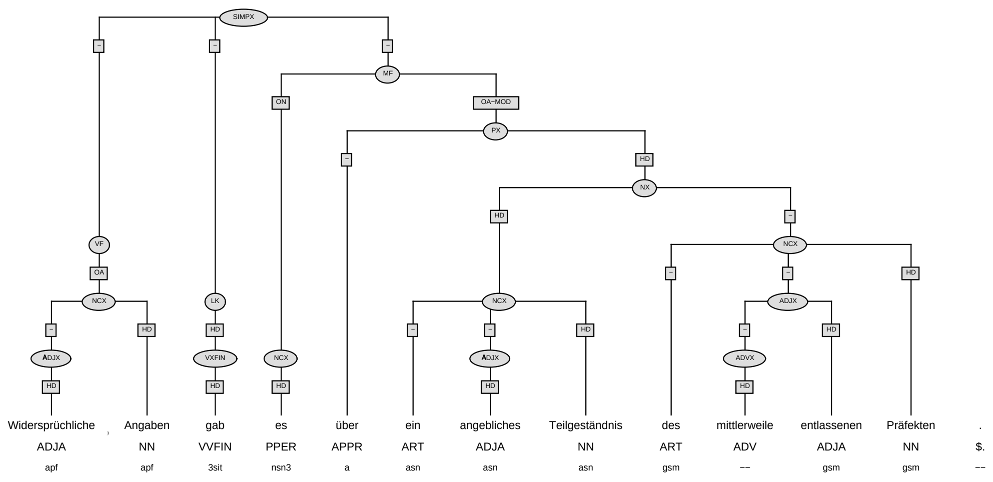


Figure 5.6: A TüBa-D/Z tree with a long-distance dependency

- If the primary edge label needs further disambiguation, e.g. if there are two `OA`s in a clause.
- If the dependency relation exists between two nodes, at least one of which is phrase internal and therefore carries only head or non-head information.
- If there is a dependency relation outside of a clause (`SIMPX`) in control verb constructions.

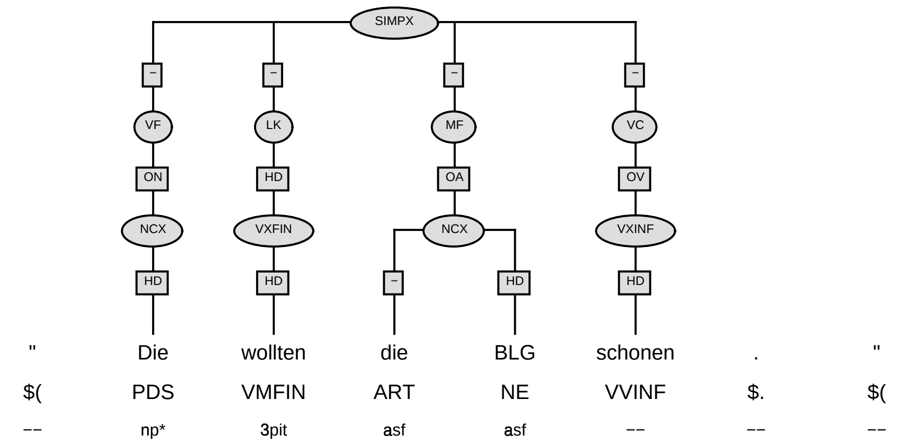
Information encoded in syntactic trees in TüBa-D/Z, namely constituency bracketing, dependency relations markers and head markers, supplies a reliable basis for transforming a TüBa-D/Z tree into a dependency structure tree (Kübler and Telljohann (2002)). Figure 5.7 provides an example of such transformation.

A complete list of edge labels of TüBa-D/Z is provided in Appendix D.2.

5.5 Conclusion

The current chapter has introduced the TüBa-D/Z treebank which represents the main data source for the research of the thesis. The main char-

A.



B.

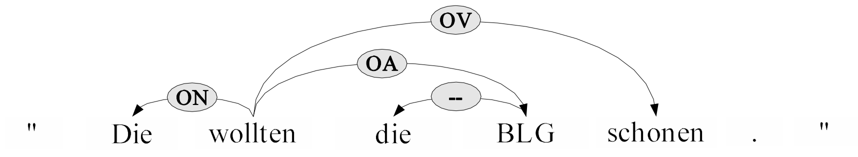


Figure 5.7: Transformation of a TüBa-D/Z tree into a dependency structure tree

acteristics of the treebank are the topological fields based annotation and absence of empty categories and of crossing branches.

Chapter 6

Morpho-syntactic tagging of German

The previous chapters have motivated the task of morpho-syntactic annotation in the context of natural language applications. A survey of existing taggers for German given in Chapter 2 has demonstrated that the morpho-syntactic annotation of German represents a particularly difficult problem. When applied to this task, state-of-the-art taggers yield results which are 10–20% lower than the results obtained with the same taggers on pure POS tagging of German. This performance is also considerably lower than the state-of-the-art performance on morpho-syntactic tagging of other languages.

The characteristics which make the task particularly difficult are the high ambiguity of the German language, a large size of the tagset used for morpho-syntactic annotation and a rather restricted data set available as training data for German.

This chapter investigates different methods for morpho-syntactic annotation of German and presents a hybrid model which achieves high performance on the given task even when trained on a restricted data set. Section 6.1 describes the development of a novel constraint-based framework that incorporates phrase-internal concord rules and phrase-external syntactic heuristics for successful resolution of morpho-syntactic ambiguity. Section 6.2 concentrates on statistical tagging approaches. First, it discusses experiments with reduced tagset techniques applied to German data. Next, it investigates the application of probabilistic phrase structure grammars (PCFGs) to the task. Section 6.3 compares the discussed models and section 6.4 presents a combined system with the rule-based and statistical modules which aims at bringing together the strengths of the methods involved. Section 6.5 concludes the chapter.

6.1 Rule-based method

Rule-based methods have been successfully applied to the tagging of different languages. Particular interest in rule-based taggers has been shown by researchers involved in tagging of highly inflectional languages, among others, Slavic languages (Hajič and Hladka (1997), Petkevič (2001)) and Turkish (Oflazer and Kuruöz (1994)). The advantages of rule-based methods that contribute to such interest consist of the ability of the rules to easily catch linguistic regularities and to incorporate a large context window, which is crucial for the correct annotation of morphological features.

The rule-based morpho-syntactic tagger described in this section is implemented as a morpho-syntactic disambiguation module based on the Xerox Incremental Deep Parsing System. The tagger is provided with ambiguously annotated text which includes all possible analyses for each token. Sequential application of constraint rules of the tagger leads to disambiguation of the input.

Below, the description of the tagger starts with the discussion of the tagger input. Next, the procedure undertaken by the tagger is illustrated by the sequential disambiguation of an example sentence. A detailed description of the two modules of the tagger, a POS disambiguator and a morphological disambiguator, follows the example. Finally, the evaluation, the error analysis and a general discussion of the tagger performance are presented.

6.1.1 Providing initial analyses with the Xerox morphological analyzer

An initial set of possible analyses for all tokens is provided to the tagger by the morphological analyzer for German developed by the Xerox Research Centre Europe (XRCE). [1] The analyzer is based on two-level morphological rules (Karttunen et al. (1992)) which are compiled into a finite-state transducer.

The analyses given by the analyzer for each token include a lemma, a POS tag and a set of relevant features. The POS tagset of the analyzer

[1] An on-line demo of an updated version of the XRCE morphological analyzer is available at `http://www.xrce.xerox.com/competencies/content-analysis/demos/german`. In the experiments described in this thesis, an earlier version of the analyzer has been used.

is based on the STTS tagset. However, some differences in the tagsets are present. These differences mainly concern pronouns and adjectives: while in the STTS tagset a distinction is made between attributive and substituting pronouns, the XRCE tagset differentiates pronominal adjectives, pronominal determiners and proper pronouns. These differences involve the closed class of lexemes and a mapping table that can be composed to provides a correspondence between the tags for tokens based on the lexemes. The list of POS tags of the analyzer together with their counterparts in the STTS tagset are presented in Appendix B.1.

Along with POS tags, the analyzer provides more fine-grained features for the analyzed tokens. Such features include morphological information (*case, number, gender, declension type, tense, mood* and *person*) and subcategories of POS (such as *country, city, first name, family name* for nouns, *degree of comparison* for adjectives, etc.). In some cases, these finer distinctions in a POS category provide useful information which allows for the precise delineation of the applicability of highly specific disambiguation rules. A full list of features used in XRCE tagset is given in Appendix B.2.

An example of a set of analyses provided by the morphological analyzer is presented in (36). It supplies analyses for tokens of sentence (37).

(36)	Der	die	+Det+Art+Pl+FMN+Gen+St+ART
	Der	die	+Det+Art+Sg+Masc+Nom+St+ART
	Der	die	+Det+Art+Sg+Fem+Dat+St+ART
	Der	die	+Det+Art+Sg+Fem+Gen+St+ART
	Der	die	+Pron+Dem+Sg+Fem+Dat+DEMPRO
	Der	die	+Pron+Dem+Sg+Masc+Nom+DEMPRO
	Der	die	+Pron+Rel+Sg+Fem+Dat+RELPRO
	Der	die	+Pron+Rel+Sg+Masc+Nom+RELPRO
	Fahrer	Fahrer	+Noun+Common+Sg+Masc+Dat+NOUN
	Fahrer	Fahrer	+Noun+Common+Sg+Masc+Acc+NOUN
	Fahrer	Fahrer	+Noun+Common+Sg+Masc+Nom+NOUN
	Fahrer	Fahrer	+Noun+Common+Pl+Masc+Gen+NOUN
	Fahrer	Fahrer	+Noun+Common+Pl+Masc+Acc+NOUN
	Fahrer	Fahrer	+Noun+Common+Pl+Masc+Nom+NOUN
	konnte	können	+Verb+Indc+1P+Sg+Past+VMFIN
	konnte	können	+Verb+Indc+3P+Sg+Past+VMFIN
	nicht	nicht	+Ptkl+Neg+PTKNEG

mehr	mehr	+Adj+Indef+Invar+INDADJ
mehr	mehr	+Adv+Common+ADV
mehr	mehren	+Verb+Imp+2P+Sg+VVFIN
bremsen	bremsen	+Verb+Indc+1P+Pl+Pres+VVFIN
bremsen	bremsen	+Verb+Indc+3P+Pl+Pres+VVFIN
bremsen	bremsen	+Verb+Inf+VVINF
bremsen	bremsen	+Verb+Subj+1P+Pl+Pres+VVFIN
bremsen	bremsen	+Verb+Subj+3P+Pl+Pres+VVFIN
.	.	+Punct+Sent+SENT

(37) Der Fahrer konnte nicht mehr bremsen.
the driver could not anymore break
'The driver could not break anymore'.

The following subsection discusses the disambiguation procedure of the tagger on the example of sentence (37).

6.1.2 Disambiguation procedure

The ambiguity introduced by the morphological analyzer is reduced by a sequential application of the rules of the two modules of the tagger: a POS disambiguator and a morphological disambiguator. Two types of disambiguation rules are used in the XIP system: syntactic heuristics and concord rules. They jointly provide an effective way to reduce morpho-syntactic ambiguity.

Concord rules are based on mutual agreement constraints between lexical nodes within one phrase, e.g. between articles and nouns within one noun phrase. They are, therefore, best suited for morphological disambiguation of lexical nodes that make up phrasal categories. Syntactic heuristics rely on constraints that a surrounding context imposes on the set of possible analyses for a given token and can, therefore, be used for both POS and morphological disambiguation.

The POS disambiguation module applies its rules in sequential order, eliminating readings that violate constraints stated in the rules. First, the relative pronoun reading (RELPRO) can be eliminated in sentence-initial position. The demonstrative pronoun reading (DEMPRO) is also ungrammatical in sentence-initial position followed by an unambiguous noun (NOUN) and a finite verb (VMFIN), since it cannot construct a phrase with the noun. This constraint relies on the theory of topological fields (Höhle (1985)), according to which only one element or phrase can occupy a Vorfeld position (i.e. the position between a clause boundary and a finite verb) in a German sentence. A finite verb reading (VVFIN) of "*bremsen*" in clause-final position can be eliminated since there is a preceding unambiguous finite verb. An imperative verb reading (VVIMP) is ungrammatical in non-clause-initial position and can also be deleted.

All the constraints mentioned above eliminate ungrammatical readings. Another possible operation of the disambiguation rules is to identify the correct analysis among the set of legitimate readings and to delete all the others. A heuristic of this type chooses an adverbial reading for "*mehr*" if the immediate left context contains a negation particle (PTKNEG).

After the application of POS disambiguation rules the sentence is further processed by the morphological disambiguation module. Example (38) demonstrates the remaining morphological ambiguity for each token of the sentence.

Concord rules in the morphological disambiguation module rely on the fact that lexical nodes within the same NP mutually constrain each other as to the set of possible readings. Application of such rules leads to elimination of all non-shared readings on tokens "*Der Fahrer*", reducing the set of possible analyses to `+Pl+Masc+Gen+St` and `+Sg+Masc+Nom+St` for both tokens. Further disambiguation of the tokens "*Der Fahrer*" is performed by a syntactic heuristic that restricts the use of genitive NPs to positions preceded by a preposition or another NP. Resolution of *person* ambiguity of the finite verb is based on the absence of a nominative pronoun with *first person* value in the clause, which allows for the elimination of the first person reading of the verb.

(38)	Der	die	+Det+Art+Pl+FMN+Gen+St+ART
	Der	die	+Det+Art+Sg+Masc+Nom+St+ART
	Der	die	+Det+Art+Sg+Fem+Dat+St+ART
	Der	die	+Det+Art+Sg+Fem+Gen+St+ART
	Fahrer	Fahrer	+Noun+Common+Sg+Masc+Dat+NOUN
	Fahrer	Fahrer	+Noun+Common+Sg+Masc+Acc+NOUN
	Fahrer	Fahrer	+Noun+Common+Sg+Masc+Nom+NOUN
	Fahrer	Fahrer	+Noun+Common+Pl+Masc+Gen+NOUN
	Fahrer	Fahrer	+Noun+Common+Pl+Masc+Acc+NOUN
	Fahrer	Fahrer	+Noun+Common+Pl+Masc+Nom+NOUN
	konnte	können	+Verb+Indc+1P+Sg+Past+VMFIN
	konnte	können	+Verb+Indc+3P+Sg+Past+VMFIN
	nicht	nicht	+Ptkl+Neg+PTKNEG

mehr	mehr	+Adv+Common+ADV
bremsen	bremsen	+Verb+Inf+VVINF
.	.	+Punct+Sent+SENT

Application of the rules described above leads to complete disambiguation of the example sentence:

(39)	Der	die	+Det+Art+Sg+Masc+Nom+St+ART
	Fahrer	Fahrer	+Noun+Common+Sg+Masc+Nom+NOUN
	konnte	können	+Verb+Indc+3P+Sg+Past+VMFIN
	nicht	nicht	+Ptkl+Neg+PTKNEG
	mehr	mehr	+Adv+Common+ADV
	bremsen	bremsen	+Verb+Inf+VVINF
	.	.	+Punct+Sent+SENT

Below, a detailed discussion of the POS and morphological disambiguation modules is presented.

6.1.3 POS disambiguation

Constraints of the POS disambiguation module are formulated as ordinary disambiguation rules (ODRs). The general ODR format presented and explained in detail in Chapter 4 is restated in (40):

(40) readings_filter = |left_context| selected_readings |right_context|.

The readings filter of a POS constraint represents a set of analyses. A constraint will apply to any node which contains a set of analyses specified in the reading filter of the constraint. The set may include any number of POS values, as well as values of other features, such as *case* or *subcategory*. References to the lemma of a token and to capitalization of a token are also possible.

Depending on the form of the readings filter, two kinds of POS disambiguation rules can be distinguished. Rules of the first kind have a single category in their readings filter. They represent general constraints about the contextual environment of tokens of a particular POS class or a subcategory. For example, infinitival particles require an infinitive in the immediate right context and the occurrence of an infinitival particle in any other context is ungrammatical.

Rules of the second type list two or more categories in their readings filter and, therefore, specify an ambiguity class to which they apply rather than a single category. Such rules describe a regularity about the contextual behavior of tokens of a particular ambiguity class and are based on differences in the distribution of analyses of tokens rather than on a regularity about the contextual environment of a single analysis. Therefore, rules of the second kind are more effective in disambiguation, but at the same time they are more prone to over-application. An example of a rule of the second kind is given in (41):

(41) conj<sub>,prep = ?[sub:∼] |?[prepart]|.

The presented rule applies to tokens with readings of a preposition and a subordinate conjunction, such as "*bis*" (= "*till, until*") and eliminates subordinate conjunction readings if the token is followed by a preposition with an article, such as "*zum*" (= "*zu dem*"). The elimination of readings is possible, since in such a context, tokens of this ambiguity class almost exclusively have a prepositional reading. However, if a token belongs to a different ambiguity class which contains either a subordinate conjunction or a prepositional reading but not both readings together, the regularity stated in the rule does not hold. Thus, subordinate conjunctions such as "*wenn*" (= "*if*") or "*ob*" (= "*whether*") can be followed by a preposition with an article:

(42)	**Wenn**	**zum**	Leben	keine	Kraft	bleibt.
	if	for	life	no	strength	remains

'If no strength remains for living.'

Also, for tokens which have a prepositional reading and which are followed by a preposition with an article, the prepositional reading is not necessarily the correct one. Example (43) demonstrates a sentence in which the word "*an*", which is originally ambiguous between a preposition and a circumposition, belongs to the former POS category:

(43)	Die	Suchmaschine	gehörte	von	Anfang	**an**	**zum**	Internet.
	the	search engines	belonged	from	beginning	-	to the	Internet

'From the outset, search engines belonged to the Internet'.

The combination of the two types of rules leads to the most efficient disambiguation: at first, general constraints about contextual distribution of POS categories eliminate readings which are ungrammatical in the given syntactic environment and then rules which refer to ambiguity classes choose a more likely analysis, given the context.

A readings filter often requires the absence of an analysis or negates a value of a feature, which further narrows down a class of considered tokens. Such extended narrowing down facilitates cautious rule application. Consider an example rule in (44):

(44) det,pron<indef:~> = ~|punct[comma],(prep)| ?[pron:~]
|adv*[prep:~],(punct[skip]),adj+[det:~],noun[pron:~]|.

The rule applies to tokens that have readings of a determiner and a pronoun and eliminates pronominal readings if the contextual constraints are satisfied. However, elimination of indefinite pronoun reading in this case is erroneous, since the occurrence of indefinite pronouns in the context specified by the rule does not result in an ungrammatical construction. Therefore, the readings filter determines that pronominal readings should have a negative value for the feature *indefinite*.

While a readings filter restricts the application of a rule to tokens that have analyses identified by the filter, fields *left_context* and *right_context* further limit the set of tokens to be considered to those tokens that occur in the environment described in the context fields. Both ambiguous and disambiguated contexts are used in POS disambiguation rules.

In the field *selected readings*, a set of readings to be output by the rule is defined. Although the selection of a single correct reading is much more efficient than the deletion of incorrect readings, it is also more insecure, since tokens in German text are highly ambiguous (5.8 analyses per token on average) and, hence, the reliable identification of a single correct reading among several possible readings based only on the context is problematic. Therefore, most rules of the POS disambiguation module operate by deleting analyses that are ungrammatical in the context specified by the rule, and only 14% of all rules disambiguate tokens by selecting contextually valid readings. A big part of that 14% of the rules rely on reference to the lemma of a token in the readings filter, which enables the additional restriction of the set of tokens. Alternatively, such selective rules apply to tokens which belong to a small ambiguity class with a complementary distribution of analyses, i.e. to tokens which have 2 or 3 possible analyses that tend to occur in different contexts.

Consider, for example, rules in (45)–(48) for the disambiguation of tokens with prepositional and postpositional readings. The first three rules (45–47) eliminate prepositional readings if the token is preceded by a noun and if the immediate right context contains a verb or a punctuation mark. Such an environment is impossible for prepositions, but is typical and necessary for postpositions. After the application of rules (45)–(47), all of the tokens that remain ambiguous between prepositions and postpositions receive a prepositional reading by rule (48).

(45) prep,postp = |noun,(punct[skip])| ?[prep:~] |(punct[skip]), verb[noun:~,adj:~,adv:~,det:~];punct[skip:~]|.

(46) prep,postp = |noun,(punct[skip])| ?[prep:~] |(punct[skip]), verb[adj,pred,noun:~,fin],?[adj:~,noun:~,card:~]|.

(47) prep,postp = |?[noun],(punct[skip])| ?[prep:~] |(punct[skip]), ?[adj,pred,noun:~,partpas],(?[verb]),?[adj:~,noun:~,card:~]|.

(48) prep,postp = ?[prep].

The rules of the POS disambiguation module are organized in layers which dictate the order of rule application. First, those rules are applied that resolve the most frequent cases of ambiguity. In this way, subsequent rules are provided with more reliable context for disambiguation.

The first layers include constraints for the disambiguation of tokens that are ambiguous between pronouns and determiners. Such constraints mostly rely on the identification of noun phrase structure for selecting determiner readings or else on the absence of a noun in the right context for eliminating determiner readings. Next, the residual ambiguity of tokens with pronominal and determiner readings is handled. For pronouns, the appropriate constraints are largely based on the contextual regularities of subclasses of pronouns, such as the ungrammaticality of a relative pronoun in sentence-initial position. For determiners, the strategy of verifying the noun phrase structure mentioned above is undertaken.

The second cluster of constraints is designed for the disambiguation of auxiliary parts of speech, such as appositions, particles and conjunctions. Since these categories belong to closed word classes, tokens are usually of a small ambiguity class and, therefore, the selection of readings instead of readings elimination is very frequent for this rule cluster. Disambiguating appositions has been partly presented above in rules (45)–(48). Appositional ambiguities further include the ambiguity between pre- and circumpositions. This ambiguity is resolved by constraints based on the left context of tokens: circumpositional readings are deleted if the token is not preceded by another preposition that can serve as a first part of a circumposition, for example, prepositions "*von*", "*zum*", "*über*". Prepositions can also be confused with different kinds of particles, such as separable verbal particles and comparative particles, with adverbs and with subordinative conjunctions. Constraint rules based on the regularities concerning the syntactic behavior of the categories involved resolve these ambiguities.

The last big group of rules deals with the disambiguation of tokens which have readings of major POS classes, such as nouns, adjectives and verbs. Among the most common ambiguities of this type are ambiguities between:

- finite verbs, infinitives and participles,
- verbs and adjectives,
- common and proper nouns,
- attributive and predicative adjectives,
- nouns and adjectives.

Intra-verbal ambiguities and verb/adjective ambiguities are resolved by constraints which check a position of the token in a clause with regard to other verbal forms and to clause boundaries.

In the disambiguation of nominal ambiguities, proper noun readings are preferred if the token is preceded by another unambiguous proper noun or by a title word, such as "*Herr*", "*Frau*", "*Professor*", etc. A proper noun reading is also chosen if the preceding word is a finite verb and if the immediate right context contains a clause-boundary punctuation. Common noun readings are preferred if the token is preceded by a cardinal, an article, a demonstrative or a possessive pronoun and is not followed by a noun. Although proper nouns can occur in such context, the majority of ambiguous words in this environment are common nouns.

Attributive and predicative adjectives ambiguities, as well as ambiguities between nouns and adjectives, are resolved by constraints based on identification of the noun phrase structure.

The order of rules in the POS disambiguation module is largely reflected by the order in which they are described above. However, for a more effective disambiguation, rules of different groups are interleaved in some cases. Such a strategy allows for a more thorough and at the same time more careful application.

The POS disambiguation module consists of approximately 400 rules. These rules are mostly unlexicalized, only 10% of them refer to lemmas in the readings filter and/or in the context fields.

6.1.4 Morphological disambiguation

The main focus of the morphological disambiguation module is on the disambiguation of the lexical nodes that make up noun phrases: nouns, determiners, adjectives and pronouns. These parts of speech represent particularly high morphological ambiguity.

The principle method that contributes to the disambiguation of NP material is based on intra-phrasal agreement: lexical nodes within the same phrase mutually constrain each other as to the set of possible readings. Consider an example in (49):

(49)	den	die	+Det+Art+Pl+FMN+Dat+St+ART
	den	die	+Det+Art+Sg+Masc+Acc+St+ART
	politischen	politisch	+Adj+Pos+Pl+FMN+NGDA+Wk+ADJA
	politischen	politisch	+Adj+Pos+Pl+FMN+Dat+St+ADJA
	politischen	politisch	+Adj+Pos+Sg+FMN+Dat+Wk+ADJA
	politischen	politisch	+Adj+Pos+Sg+FMN+Gen+Wk+ADJA
	politischen	politisch	+Adj+Pos+Sg+Masc+Acc+St+ADJA
	politischen	politisch	+Adj+Pos+Sg+Masc+Acc+Wk+ADJA
	politischen	politisch	+Adj+Pos+Sg+Masc+Gen+St+ADJA
	politischen	politisch	+Adj+Pos+Sg+Neut+Gen+St+ADJA
	Direktoren	Direktor	+Noun+Common+Pl+Masc+NGDA+NOUN

Each token in the phrase is many times ambiguous: note that the tags `FMN` and `NGDA` stand for *any gender* and *any case*, correspondingly, and are to be expanded into a set of four analyses with different gender values (`FMN`, `Fem`, `Masc`, `Neut`), for `FMN`, and into a set of five analyses with different case values (`NGDA`, `Nom`, `Gen`, `Acc`, `Dat`), for `NGDA`.[2] The union of the sets of analyses of the tokens covers all possible readings except for `+Sg+Masc+Nom`, `+Sg+Masc+Dat` and `+Sg+Neut` any case but genitive. However, the distribution of analyses differs between tokens. Imposing a constraint that requires identical values of *gender*, *number* and *case* for all tokens which belong to the same noun phrase leads to the complete disambiguation of the determiner and the noun:

[2]The analyses `FMN` and `NGDA` are left after the expansion of the tags, since some lexical tokens have underspecified values for these features. For example, a noun "*Gehörlose*" ("*deaf people*") is unspecified as for the gender value, and there are contexts that do not provide sufficient information for the resolution of case value, in which case the value is left underspecified.

(50)	den	die	+Det+Art+Pl+Masc+Dat+St+ART
	politischen	politisch	+Adj+Pos+Pl+Masc+Dat+Wk+ADJA
	politischen	politisch	+Adj+Pos+Pl+Masc+Dat+St+ADJA
	Direktoren	Direktor	+Noun+Common+Pl+Masc+Dat+NOUN

An ambiguity concerning the *declension type* of the adjective still remains. In German, word forms for adjectives and determiners can be classified as belonging to either weak or strong declension types.[3] For example, all forms of the definite determiner "*der*" belong to the strong declension type, while the paradigm of the indefinite determiner "*ein*" is split between weak and strong forms. In addition, some nouns, in particular those derived from adjectives such as "*Gehörlose*", also exhibit a distinction between weak and strong forms.

If determiners co-occur with adjectives and nouns in the same NP, adjective and noun agree in declension class, whereas the declension value of the determiner is the opposite. Thus, the adjective in the phrase "*den politischen Direktoren*" must be of a weak declension type (`Wk`), since the determiner "*den*" has a value *strong* (`St`) for this feature. This constraint disambiguates the tokens in the noun phrase completely.

The intra-phrasal agreement constraints discussed above are incorporated in the concord rules of the morphological disambiguator module. Such rules are implemented as double reduction rules (DRRs) discussed in Chapter 4.

Example (51) illustrates a concord rule stated in the form of a DRR. This rule eliminates all readings of adjectives and nouns that do not match. The pattern matching algorithm of the XIP System ensures the non-deterministic application of the rule to each adjective that precedes a noun in a left-to-right fashion.

(51) $|\text{adj}^*, \text{a-mod}^*, \text{adj}\#1, \text{a-mod}^*, \text{adj}^*, \text{noun}\#2| \Rightarrow (\#1[\text{agr}] :: \#2[\text{agr}])$.

The category `a-mod` in the rule stands for optional AP material such as adverbs and cardinal numbers. For readability, the category is omitted in the following rules; however, in the actual rules of the morphological module, `a-mod` material always precedes `adj`.

The condition on the right-hand side of the rule (with the identity operator ": : ") enforces strict identity of agreement features between the adjective and the noun, with agreement consisting of the gender, number and

[3]For a comprehensive study of distributional properties of weak and strong forms in German NP see Zwicky (1986).

case features for each node. Therefore, the rule has the effect of eliminating all readings of adjective and noun sequences with conflicting agreement features. However, if the nodes in question have no common readings to start with, then no readings are eliminated.

The rule in (52) accounts for the distinct declension type values required for the contextually valid patterns of determiners and adjectives that have been discussed above.

(52) |det#1, adj*, adj#2, adj*, noun| ⇒ (#1[agr] : : #2[agr]) & (#1[decl] ∼: #2[decl]).

If there is no determiner in front of a sequence of an adjective and a noun, then all weak readings of the adjective and the noun should be eliminated. This is handled by rule (53):[4]

(53) |?[det: ∼,a-mod: ∼], adj*, adj#1, adj*, noun#2| ⇒ (#1[agr] : : #2[agr]) & (#1[decl: St]) & (#2[decl: St]).

Rules (52) and (53) illustrate another feature of the expressivity of DRRs: the constraint on the right-hand side of the DRR may contain any combination of Boolean operators (disjunction, conjunction and negation of features) that can be expressed in the system. To force distinctness of declension values the negated equality operator ∼: is used.

The full expressivity of DRRs makes it possible to state conditions on contextually valid morphological readings as succinctly as possible. This is one of the main advantages of the present approach over previous frameworks for morphological disambiguation.[5] While the framework of constraint grammar used by Voutilainen (1995*a*) permits Boolean constraints, it lacks an equality operator and the use of variables over features on adjacent nodes. This, in turn, means that constraints cannot be generalized, but have to be stated in a case by case fashion. While this may be tolerable for languages like English, it will lead to an explosion of rules for languages like German with richer morphological paradigms.

Hajič et al. (2001) and Oflazer and Tür (1997) do not consider agreement phenomena of the sort treated here. Therefore, it is difficult to tell whether the syntax of their disambiguation rules is rich enough to accommodate the same level of generality provided by the DRRs.

[4]For readability the rule has been simplified by leaving out optional NP/AP material, such as adverbs and cardinal numbers.

[5]Petkevič (2001) seems to envisage rules similar to the ones used in the described tagger. However, he does not provide any formal specification or semantics for disambiguation rules, which makes a precise comparison difficult.

A wide range of intra-phrasal agreement regularities is covered by concord rules: prenominal agreement (with determiners, adjectives, cardinals, measure phrases, participial premodifiers, etc.), determiner-noun agreement in noun phrases with embedded constructions, case agreement with prepositions and agreement in complex proper names. Moreover, such regularities as subject-verb agreement and simple nominal coordinations are also stated in concord rules.

However, concord rules do not provide sufficient ground for resolving the ambiguity completely. Since case syncretism is a usual phenomenon for German, many noun phrases retain more than one valid analysis after the application of concord rules. Moreover, intra-phrasal concord rules only apply to noun phrases which consist of more than one lexical node that exhibits inflectional morphology. These rules, therefore, do not apply to single-element NPs such as relative or personal pronouns. In order to disambiguate such single-element NPs and to further disambiguate complex NPs, the morphological disambiguator employs syntactic heuristics stated in the form of ordinary disambiguation rules (ODRs)[6]. One of the most effective syntactic heuristics of the disambiguator is to retain only the nominative case reading for an NP if that NP is the only candidate for being the subject (i.e. it is the only NP in a finite clause or the only NP with a nominative reading in a finite clause). Table 6.1 provides an overview of some of the more effective heuristics implemented in the rule-based tagger. For each heuristic, Table 6.1 shows which case value is retained or eliminated. The numbers in Table 6.1 indicate the approximate percentage of ambiguous NPs that received a unique reading after the application of the heuristic.[7] For expository purposes, the informal rendering of the heuristics in Table 6.1 leaves out many of the necessary contextual restrictions. A more detailed discussion of the contextual restrictions for these heuristics is provided in the Appendix E.

Syntactic heuristics and concord rules are freely mixed in the tagging system. In fact, interleaving of the two rule types as well as cyclic application of rules is often necessary. For example, as a result of an earlier application of concord rules, there often remains only one head noun that can be nominative (but does not necessarily have only one possible analysis for case). Since every finite clause requires a subject, i.e. a noun phrase in nominative case, the non-nominative readings for this one noun can then be

[6]For a description of the format of ordinary disambiguation rules, see Chapter 4.

[7]Such approximation is very hard to provide for large test data sets. Numbers in Table 6.1 are estimated on a smaller test data set used in the experiments described by Hinrichs and Trushkina (2002).

Description of a Syntactic Heuristic	Case value	Percentage
The NP is the only one in a finite clause (then it is a single candidate for subject).	Nom	16.57%
A noun with feature `City` or `Country` is preceded by a preposition "*in*".	Dat	4.07%
Eliminate `Nom` reading on ambiguous NPs if there is a non-ambiguous `Nom` NP in a clause (with no coordination or comparison).	¬ Nom	3.66%
The NP is an argument of a copula verb.	Nom	3.26%
A nominative reading does not agree with a finite verb in number.	¬ Nom	2.16%
The NP is preceded neither by a preposition nor by another NP.	¬ Gen	1.62%
The NP is a non-initial NP in a Vorfeld position in a *verb-second* clause.	Gen	1.21%
The NP is a complement of a `zu`-infinitive.	¬ Nom	1.09%

Table 6.1: Syntactic Heuristics

eliminated by a syntactic heuristic. This reduction of readings on the head noun can, in turn, lead to the further reduction of the other lexical nodes (e.g. preceding determiners and adjectives) that belong to the same noun phrase.

The morphological disambiguation module consists of approximately 1000 rules. This count includes the repetitive application of rules. About 250 of the tagger rules are concord rules. Such a large amount of rules is due to the contextual differences stated in the constraints. Similar to the disambiguation system of Voutilainen (1995*b*), the tagger described above employs a rather small set of syntactic generalizations. What makes the rule set expand significantly is the specification of the context and the internal structure of phrases which can be rather heterogeneous.

6.1.5 Evaluation

Performance of the tagger has been evaluated on 8949 manually annotated tokens from the TüBa-D/Z treebank. For the evaluation, the analyses provided by the tagger have been mapped into the corresponding tags of the treebank format.

Table 6.2 provides the evaluation of the rule-based disambiguation model. The first line represents a baseline for the model performance: it is calculated as the performance of the morphological analyzer. The next

module	preci-sion	recall	F-me-asure	LE	DE	ambiguity tokens	R1	R2
Analyzer	16.90%	98.06%	18.84%	100%	0%	66.61%	8.21	5.8
POS	23.19%	97.56%	37.47%	78.42%	21.58%	56.33%	6.69	4.21
Morph.	48.27%	96.45%	64.34%	53.70%	46.30%	29.27%	4.41	2.60
+ adding analyses	50.54%	96.59%	66.36%	47.69%	52.31%	29.06%	4.14	1.91

Table 6.2: Evaluation of the rule-based disambiguation model

two lines stand for the POS and morphological disambiguation modules, respectively.

The table contains numbers for *precision, recall* and *f-measure*, as well as the percentage of ambiguous tokens in the test data together with 2 ambiguity rates. Column `R1` presents the average ambiguity rate of ambiguous tokens only, whereas column `R2` provides the average ambiguity rate for all tokens. To simplify comparison with the results obtained by other researchers, the formulas described in the earlier literature (see Hajič et al. (2001)) are used:

$$Precision = \frac{\#Tokens\ with\ a\ correct\ tag}{\#Analyses\ generated}$$

$$Recall = \frac{\#Tokens\ with\ a\ correct\ tag}{\#Tokens\ in\ data}$$

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

Following Volk and Schneider (1998), the errors made by the tagger are split into lexical errors (LE; column 5) and disambiguation errors (DE; column 6). Lexical errors are caused by the morphological analyzer: the correct analysis is not present among the set of analyses assigned by it. Disambiguation errors are proper errors of the model: the correct analysis has been deleted during rule application.

As column 5 (LE) of Table 6.2 shows, the majority of the errors are lexical errors which are due to the deficiency of the morphological analyzer. A big part of such errors concerns foreign material and proper names – these lexemes do not contain necessary morphological clues for successful identification of correct analyses and are often confused with other parts of speech: common nouns, adjectives and verbs.

An attempt was made to decrease the initial error rate caused by the morphological analyzer. Unknown words were looked up in the external list of foreign material words and proper names extracted from the training data, as well as in the list of personal names extracted from the online newspapers of the period of 2001-2003. If the lists contained the unknown word, the analyses provided by the morphological analyzer were replaced with the analyses from the lists.[8] The performance of the rule-based disambiguation model run on such an extended input is shown in the last line of Table 6.2.

Another major source of lexical errors is the confusion between adverbs and predicative adjectives, the annotation of which is often guided by semantic criteria and constitutes a difficult case for the morphological analyzer. Together with errors caused by rule over-application, lexical errors lead to decreased recall.

The tagger successfully reduces the ambiguity of tokens, leading to full disambiguation of 70% of tokens. Evaluation of the tagger performance on only fully disambiguated tokens has demonstrated an accuracy of 97.62%. The overall ambiguity rate of tokens decreased from 5.8 analyses to 1.91 analyses per token, whereas the drop in accuracy amounts to only 1.5%. However, the remaining ambiguity of 29.06% of tokens results in a decrease in the precision of the model (50.54%). The disambiguation rules are mostly eliminative in nature and in many cases the surrounding context does not provide enough evidence for the deletion of an analysis as ungrammatical.

Manual analysis of the rule-based model's output has shown that a big part of the remaining ambiguity cases represent ambiguity which is hard or impossible to resolve based only on the context and ambiguity class evidence and which requires knowledge of semantic and/or pragmatic information for successful resolution. Such cases include, among others:

- Nominative/accusative ambiguity of tokens which constitute noun phrases:

(54) In spektakulären Fällen haben **gefeuerte Ältere** in den USA
In sensational cases have fired seniors in the USA
ihr Unternehmen wegen Altersdiskriminierung verklagt.
their companies against age discrimination sued
'In the USA, fired seniors sued in sensational cases their companies against age discrimination'.

[8]Analyses were added to 1.74% of tokens in test data.

If after the application of concord rules and heuristics described above, two (or more) NPs remain ambiguous between nominative and accusative cases, other syntactic characteristics, such as word order, cannot provide sufficient evidence for the disambiguation of the NPs and semantic criteria are needed. Therefore, such cases are left ambiguous by the tagger.

- Accusative/dative ambiguity of tokens which constitute prepositional phrases:

(55) **In Plattenläden** hängt ihr Plakat.
In record shops hangs her poster.

The case value of a prepositional phrase depends on semantic characteristics of the governing verb: verbs with a movement component in the meaning, such as "*hang up*", require an accusative PP, whereas static verbs, such as "*dangle*", take dative PPs. Since such information about verbs is unavailable for the tagger, the accusative/dative ambiguity of prepositional phrases is kept unresolved.

- Ambiguity between adjectives and adverbs:

(56) Es soll zum Wochenende sogar **teilweise** Schauer
It should at weekend even partial/partly rains
geben.
be.
'It will be partly rainy over the weekend'.

- Ambiguity between subjunctive and indicative mood of verbs:

(57) Starke Männer **weinen**.
Strong men cry/would cry.

The last two cases represent other kinds of ambiguity that cannot be resolved based on contextual information.

Table 6.3 demonstrates the distribution of errors made by the rule-based tagger among the morpho-syntactic features. In the second column (POS) the percentage of errors involving POS categories is presented. Columns 3–6 provide the distribution of errors that occur (only) in one of the morphological feature values, while the values of the other morphological features and

module	POS	case	number	gender	person	tense	mood
Analyzer	53.63%	3.52%	7.27%	2.72%	0.90%	7.27%	0.90%
POS	65%	3.26%	5.71%	2.14%	0.71%	5.71%	0.71%
Morph.	44.93%	17.39%	5.80%	2.41%	1.45%	3.86%	0.48%
+ adding analyses	42.21%	18.09%	5.02%	4.52%	0.51%	4.02%	0.50%

Table 6.3: Error analysis for the performance of the rule-based tagger

of the POS category are correct. Errors involving more than one morphological feature are not considered in the error analysis of Table 6.7, since in such cases it is unclear which feature is ultimately responsible for the error.

6.2 Statistical methods

Statistical taggers currently represent the most widely used tool for POS annotation. The popularity of statistical taggers has to do with their high performance and speed as well as with the simplicity of their application. In this section, the application of statistical methods to morpho-syntactic tagging is investigated.

In all the experiments with statistical taggers reported here, the TüBa-D/Z treebank data has been used. The tagset of the treebank has been described in Chapter 5. It includes 1317 distinct tags and uses STTS tags combined with morphological features for *case, number, gender, tense, mood* and *person.*[9]

The experiments have been performed on three data sets of different size. The data sets include training data, tuning data and test data. Tuning set is the same for all experiments, whereas the training and testing data of smaller sets represent a subset of training and testing data of the larger sets. Table 6.4 provides statistics for the three data sets.

No.	Name	train	tune	test
1.	50k	51 288	5 854	8 949
2.	100k	104 049	5 854	11 361
3.	150k	155 042	5 854	17 179

Table 6.4: Data sets used in the experiments with statistical models

[9]A list of TüBa-D/Z tags is presented in Appendix C.

6.2.1 Trigram models

Among different statistical taggers, trigram model taggers have been shown to provide the optimal trade-off between accuracy and the size of the training set involved. Since the amount of available morphologically annotated data for German is rather restricted, trigram models represent the best alternative for supervised morpho-syntactic tagging of this language. Below, the experiments with the morpho-syntactic tagging of German with a trigram model are described.

It is well known that significant expansion of the tagset leads to decrease in performance for all tagging frameworks. Therefore, the application of methods which aim at alleviating the undesirable effect of the incorporation of morphological information into the tagset is also discussed.

TnT experiments

In the trigram experiments, the TnT tagger has been used. Brants (2000) has shown that the tagger achieves the same state-of-the-art performance of 96.7% accuracy when applied to pure POS tagging of both English and German data. Given that the German language has a more flexible word-order and a higher ambiguity than English, the fact that TnT provides equal results for both languages demonstrates a high suitability of the tagger for German. This makes TnT a particularly appropriate choice in the experiments with morpho-syntactic tagging of German.

When trained and tested on the three TüBa-D/Z data sets, TnT achieves a baseline accuracy of 74.97%, 80.27% and 82.68%, correspondingly.

data	full tagset	POS only
50k	74.97%	93.39%
100k	80.27%	95.79%
150k	82.68%	96.50%

Table 6.5: Evaluation of the pure TnT model

As compared to the accuracy of TnT reported by Brants (2000), the achieved accuracy is rather low. This difference in performance is due to expansion of the tagset and to a smaller amount of training data. To estimate the influence of the training data size reduction, experiments with pure POS tagging have been performed on the same data. As Table 6.5 demonstrates, for the STTS tagset, the tagger achieves a maximum accuracy of 96.50%.

This number is comparable to the accuracy of 96.70% reported by Brants (2000), who trained TnT with the STTS tagset on the NeGRA newspaper corpus with approximately 320 000 tokens, i.e. a training corpus more than twice as big as the largest TüBa-D/Z training corpus used here.

data	accuracy
50k	80.61%
100k	86.88%
150k	87.48%

Table 6.6: Evaluation of the TnT model with a back-up lexicon

The first improvement in the performance of TnT on morpho-syntactic annotation can be gained if TnT is provided with a back-up lexicon which contains the set of all possible morpho-syntactic analyses for unknown words. This restricts the search space of the tagger and avoids errors caused by assigning invalid tags to tokens. The back-up lexicon has been produced by mapping the analyses provided by the Xerox morphological analyzer into the treebank tagset format. The obtained improvement is presented in Table 6.6. On different data sets it amounts to 4.8–6.61%.

Table 6.7 demonstrates the distribution of errors among the morpho-syntactic features for the experiments with TnT augmented with the back-up lexicon. Isolating the errors of morphological features in this way demonstrates that the main source of errors made by TnT are case and POS categories.

Further improvements to the performance of TnT can be obtained by either increasing the size of the training data or by reducing the tagset. Considering the amount of manual effort necessary, the former strategy is not feasible in practice. The latter strategy has been advocated by Tufiş (2000) and by Dienes and Oravecz (2000). Below, experiments with the tagging method based on the reduction of the tagset are described.

data	POS	case	number	gender	person	tense	mood
50k	35.85%	37.00%	1.90%	6.51%	0.75%	0.86%	0.40%
100k	28.79%	43.02%	1.81%	5.70%	1.28%	0.34%	1.41%
150k	31.47%	41.05%	1.63%	5.58%	0.79%	0.09%	1.02%

Table 6.7: Error analysis for the performance of TnT with a back-up lexicon

Experiments with a reduced tagset

Several experiments in line with the ideas of Tufiş (2000) and Dienes and Oravecz (2000) have been performed on German data. The main goal of tagset reduction approaches consists of optimizing the balance between the size of the tagset used for tagging and the size of the training data set. Tufiş (2000) has shown that for Romanian, values of many features used in tags are easily recoverable from the lexical information and can therefore be omitted from the original tagset for the intermediate tagging. In the same manner, some values of a feature can be merged together if the information is recoverable. For example, the original tagset used in the experiments of Tufiş (2000) includes a feature *object on verbs* with values `I` (no object or indefinite object), `D` (definite object) and `2` (incorporated second person object). However, the distinction between classes `I` and `2` is recoverable, since the relevant information is contained in the form of the lexemes: no token in Romanian is ambiguous between these two classes and therefore, the value can be reconstructed by a simple procedure of looking up the lexeme analyses in a lexicon. Moreover, lexemes of these classes behave in the same way syntacticly and thus do not provide any contextual clues for the disambiguation of other tokens. Hence, the classes can be merged, which results in collapsing those tags that differ only in the value of this feature in one generalized tag.[10] Applying this procedure recursively to all features of the tags, the tagset size can be significantly reduced.

A formal description of the algorithms of Tufiş (2000) and Dienes and Oravecz (2000) is presented in Appendix G.1 and Appendix G.2, correspondingly. Informally, they differ in the direction of the tagset reduction: the tiered tagging (T-tagging) approach of Tufiş (2000) sequentially reduces the original tagset top down, whereas the bottom-up tagset design (BUTD) of Dienes and Oravecz (2000) maximally reduces the original tagset without loss of information and then re-introduces morpho-syntactic features to expand the reduced tagset to a set of tags that exhibit sufficient distributional cues for the tagger.

Both algorithms have been implemented for application to the German data. Table 6.8 shows the results of applying these two algorithms to the data set 1 (50k). A straightforward implementation of the BUTD approach fails to reach the baseline set by TnT with the maximally reduced tagset (72.33%). Experiments with expanded tagsets whose classes respect differences in major word classes yield a maximal, but still below baseline, result of 78.19%.

[10]The example is taken from Tufiş (2000).

	tune set		test set	
	pure	back-up	pure	back-up
TnT	75.09	80.74	74.97	80.61
BUTD				
straightforward	72.51	72.53	72.15	72.33
expanded tagsets	76.23	78.49	75.43	78.19
T-tagging				
straightforward (10% info loss)	82.71	83.58	81.84	83.13
Case, Number and Person only	83.59	83.02	83.26	83.01
no Case, Number and Person	85.39	85.58	84.03	84.94

Table 6.8: Experiments with reduced tagsets

The T-tagging approach is able to outperform the TnT baseline. The standard margin of 10% information loss set by the algorithm of Tufiş (2000) yields 83.13% for German data. However, this higher result is achieved at the cost of losing information contained in the original tagset. The algorithm described by Tufiş (2000) leaves out an attribute from all the tags in the reduced tagset if the elimination results in limited percentage (less than 10%) of words becoming ambiguous after recovering the original tags. The recovering process is lexicon driven: it is equivalent to computing the intersection between the set of original tags that map into the reduced tag assigned to the token and the set of analyses specified for the token in the lexicon. For resolving the remaining 10% ambiguity, the algorithm relies on hand-crafted rules. However, for German this strategy turns out not to work: 10% information loss results in the elimination of all morphological features on major word classes, e.g. finite auxiliary verbs, articles, proper nouns as well as personal, demonstrative and relative pronouns and determiners. This type of information is not recoverable by hand-crafted rules, thus limiting the utility of the tagged output.

It is important to note that Tufiş (2000) reports a much lower ambiguity rate for Romanian – approximately 1.7 readings per token – compared to approximately 5.8 readings per token prior to part-of-speech-tagging and morphological disambiguation for the German test corpus that has been used for evaluation. Moreover, compared to German, case syncretism in Romanian seems to follow much more systematic patterns across nominal paradigms: nominative forms consistently coincide with accusative forms, whereas corresponding forms of two other cases, dative and genitive, represent another identical pair. Romanian, therefore, lends itself to a straightforward reduction of the full tagset. The same is true for Hungarian: the ambiguity rate is even lower than for Romanian – approximately 1.3 readings per token – and case values are easily recoverable from lexical information, as collapsing of 21 original cases to 3 case distinctions (nominative, accusative and other) has shown (Tufiş et al. (2000)). For German, such a compact merging of cases does not seem possible, since case syncretism varies from paradigm to paradigm so that only a set of subregularities can be induced. Table 1.1 from the introduction repeated here in Table 6.9 demonstrates that conflation of nominal case forms follows different patterns depending on the paradigm: all case forms are identical for nouns like "*Blume*", nominative form is opposed to all other case forms for nouns like "*Bär*", dative form is opposed to all other case forms for nouns like "*Bilder*", etc. Additionally, case syncretism represents a different picture for pronominal forms, for adjectival forms and for determiner forms, which leads to a further complication. This lack of strong regularities in the German case system violates the requirements of the T-Tagging compaction algorithm, which results in the poor performance of the reduced tagset approaches on German data.

Nominative	Genitive	Dative	Accusative
Blume	∼	∼	∼
Bär	∼en	∼en	∼en
Bilder	∼	∼n	∼
Vater	∼s	∼	∼
Name	∼ns	∼n	∼n

Table 6.9: An example of German nominal case syncretism

It is instructive to profile the T-tagging approach by comparing the retention of different sets of crucial morphological features. If only case and number information is retained on nominal categories and only number and person information is retained on finite verbs, the result is a lower accuracy (83.01%) than if only these features are deleted from the tagset and all other morphological features are retained on these and on all other word classes (84.94%). *Case, number* and *person* are generally regarded as the most basic morphological features to be included in any tagset with inflectional information. The fact that this feature set is more difficult to accommodate than all other morphological features combined points at a crucial deficiency in the underlying model for the task at hand.

Limitations of n-gram models

N-gram taggers such as TnT consider only sequences of **n** words and their candidate tags, i.e. very local contexts, as the basis for determining the most likely sequence of tags for the sentence. This Markovian assumption proves harmful for decisions that crucially require larger context windows. *Case, person* and *number* information is precisely of this nature, since successful disambiguation needs to rely on genuinely syntactic phenomena such as subject-verb agreement, valency of main verbs, and morphological features of other nominal elements in the sentence. While at first sight it seems that *case* and nominal *number* can be resolved within a small context window via phrase-internal agreement, experiments with the rule-based tagger described in the previous section show that in a lot of cases use of syntactic heuristics which take into account distant features is necessary. The most prominent example of this kind is illustrated in (58). The example is taken from the error analysis of TnT performance on test data.

(58)	Die	Frage	nach	der	Form	beantwortet	er	dann	auch	so:
	The	question	about	the	form	answers	he	then	also	in this way

‘He answers the question about the form in this way:’

The sentence contains an unambiguous nominative pronoun “*er*” and a noun phrase “*die Frage*”. The latter NP represents a common nominative-accusative ambiguity. TnT wrongly assigned nominative case to the tokens “*die Frage*” even though the combination of the nominative and accusative NPs in a clause is much more likely than a pattern of two nominative NPs. This deficiency in the statistical model used by the TnT tagger is due to its extremely local context window.

The most widely used probabilistic models that can incorporate more global structural information are *probabilistic context-free grammars* (PCFGs). The conjecture that PCFGs yield better results is confirmed by experiments with the PCFG-parser LoPar designed by Schmid (2000). These experiments are described in the following section.

6.2.2 PCFG model

Probabilistic Phrase Structure Grammars[11]

A Probabilistic Phrase Structure Grammar is an extension of a context-free grammar, in which each rule is associated with a probability. Formally, a

[11]The introduction into PCFGs presented here is based on the introduction of Manning and Schütze (1999).

PCFG grammar G is a five-tuple $< W, N, S, R, P >$, where

- W is a set of terminal symbols $w^1, ..., w^w$,
- N is a set of non-terminal symbols $N^1, ..., N^\nu$,
- S is a start symbol, $S \in N$,
- R is a set of rules, each of which is of the form $N^i \to \zeta^j$, where ζ^j is a string over $W \cup N$,
- P is a corresponding set of probabilities on rules such that

$$\forall i \ \sum_j P(N^i \to \zeta^j) = 1.$$

The probability of a parse is computed as the product of the probabilities of all rules applied in the parse. The probability of a sentence is the sum of probabilities of all possible parses of the sentence:

$$P(w_{1m}) = \sum_{t_{w_{1m}}} P(t)$$

where $t_{w_{1m}}$ is a parse of the sentence w_{1m}.

Since the calculation of the probabilities of all possible parses for a sentence is very inefficient, the total probability of a sentence can be calculated by dynamic programming algorithms.

The *inside algorithm* is based on the *inside probabilities*. The probability of a string w_{1m} is calculated as the inside probability of the root node:

$$P(w_{1m} \mid G) = P(S \overset{*}{\Rightarrow} w_{1m} \mid G) = P(w_{1m} \mid S_{1m}, G) = \beta_1(1, m)$$

where notation $S \overset{*}{\Rightarrow} w_{1m}$ stands for generating a string w_{1m} starting from node S. S_{1m} means that S spans positions 1 through m. β_S is the inside probability of the root node.

The inside probability of a node N^j that spans positions p through q is defined as

$$\beta_j(p, q) = P(w_{pq} \mid {N^j}_{pq}, G)$$

and is calculated inductively as

$$\beta_j(p, q) = \sum_{r,s} \sum_{d=p}^{q-1} P(N^j \to N^r N^s) \ \beta_r(p, d) \ \beta_s(d+1, q)$$

The base for the induction is the probability of a leaf node $\beta_j(k,k)$ which is equal to the probability of a rule $N^j \rightarrow w_k$:

$$\beta_j(k,k) = P(w_k \mid {N^j}_{kk}, G) = P(N^j \rightarrow w_k \mid G)$$

The probability of a string can be also found with the *outside algorithm* using the *outside probabilities* of pre-terminal nodes:

$$P(w_{1m} \mid G) = \sum_j \alpha_j(k,k)\ P(N^j \rightarrow w_k)$$

where α_j is the outside probability of N^j.

The outside probability of the root node S that spans the whole string is

$\alpha_1(1,m) = 1$
$\alpha_j(1,m) = 0$ for j $\neq 1$

The outside probability of an internal node is defined inductively as

$$\alpha_j(p,q) = [\sum_{f,g \neq j} \sum_{e=q+1}^{m} \alpha_f(p,e)\ P(N^f \rightarrow N^j N^g)\ \beta_g(q+1,e)]$$

$$+[\sum_{f,g} \sum_{e=1}^{p-1} \alpha_f(e,q)\ P(N^f \rightarrow N^g N^j)\ \beta_g(e,p-1)]$$

The *inside-outside algorithm* computes the joint probability of a sentence and a node spanning from position p to position q as

$$P(w_{1m}, N_{pq} \mid G) = \sum_j \alpha_j(p,q)\ \beta_j(p,1)$$

This formula describes the probability of a sentence, analyses of which use a constituent N^j covering words p through g.

The most likely parse of a sentence can be found with slight modifications of the algorithms described above. The algorithms are adapted to find the arguments that provide the maximum of functions instead of the sums of the function values. The rule that yielded this maximum is recorded. The most likely path can be identified by backtracking the recorded values.

Tagging with PCFGs

The PCFG parser LoPar is a particularly suitable tool for the experiments on morpho-syntactic tagging since it provides dedicated Viterbi parsing and tagging modes. The corresponding modes give different results when applied to the task of morpho-syntactic annotation with large tagsets. In Viterbi parsing mode, LoPar computes the most probable tree structure for a given input string. This mode can be used for part-of-speech tagging by outputting the sequence of part-of-speech tags that appear as preterminal nodes of the most likely tree. In tagging mode, the best tag sequence is computed independently of the most likely parse tree. In this mode the best tag sequence is defined as the sequence of those tags that yield the maximal product of the inside and outside probabilities among the candidate tags for a given word. A more formal comparison between the two modes is given below.

As has been pointed out above, the joint probability of a string $\mathtt{w}_{1m}$ and a node $\mathtt{N}_{pq}$ spanning from string position `p` to `q`, given a grammar `G`, can be computed by the formula in `A1`.

$$P(w_{1m}, N_{pq} \mid G) = \sum_j \alpha_j(p,q)\beta_j(p,q) \tag{A1}$$

In LoPar tagging mode, the best tag sequence is defined as the sequence of those tags that yield the maximal product of the inside and outside probabilities among the candidate tags for a given word. Each tag is therefore computed by the following formula:

$$\arg\max_j \alpha_j(k,k)P(N^j \to w_k) \tag{A2}$$

`A2` is an instantiation of `A1` for preterminal nodes spanning from `k` to `k` with the sum operator replaced by *argmax*. $P(N^j \to w_k)$ denotes the inside probability of the tag.

In Viterbi parsing mode, LoPar computes the sequence of part-of-speech tags that appear as preterminal nodes of the most likely parse tree. What makes a direct comparison to the computation of tag sequence with the Viterbi parsing mode difficult is that computation of the most likely parse tree is usually performed in terms of inside probabilities only. This amounts to instantiating `p` and `q` in formula `A1` to `1` and `m` and to a Viterbi-style computation of maxima, rather than sums, for inside probabilities. Below, the probabilities of the best inside analyses computed this way are referred to as vit_β. The probability of the best parse tree is then computed as $vit\_\beta_1(1,m)$.

Alternatively, one can instantiate `p` and `q` in `A1` to shorter subsequences of the input string and determine the most likely tree by a Viterbi-style computation of both inside and outside probabilities. In particular, one can instantiate `p` and `q` to a single `k` and dynamically compute maxima (rather than sums) for outside probabilities of a preterminal node. Below, the probabilities of the best outside analyses computed in this way are referred to as vit_α. The probability of the best parse tree is then computed as $vit\_\alpha_j(k,k)$ $P(N^j \to w_k)$. The best tag $\mathtt{N}^j$ for a given word $\mathtt{w}_k$ inside the best parse tree can be computed by the following formula:

$$\arg\max_j vit\_\alpha_j(k,k) P(N^j \to w_k) \tag{A3}$$

Formula `A3` for Viterbi parsing constitutes a specification in terms of outside probabilities that directly parallels the main components of the formula in `A2` which is used in LoPar's tagging mode. Stating the relevant formulas in this way should help clarify the similarities and differences between the two LoPar modes.

Since LoPar's tagging mode computes the most likely tag sequence independently of the most likely parse tree, the best tag for a given word will be identified as that tag which provides the best balance between the likelihood of the tag itself and the likelihood of the surrounding syntactic context. Since this choice is determined for each tag independently of all the others, the resulting tag sequence may well differ from the tag sequence of the most likely parse tree. Viterbi parsing has as its goal the construction of a single tree that provides the best balance between the individual probabilities of the candidate structures of its parts. In order to arrive at this balance, the most likely analysis of an individual phrase or word may not be incorporated into the most likely tree if the outside probability for that analysis is much inferior to the outside probabilities of other candidate analyses for the same word or phrase. Thus, Viterbi parsing optimizes global structure, not individual substructures. Since part-of-speech tagging is of a more local nature compared to full sentential analysis, it should therefore be expected that independent maximization of tags for individual words should perform better than global maximization of full sentential structures.

The experiments with both Viterbi parsing and tagging modes demonstrate an average improvement of the tagging mode performance over the performance of the parsing mode that amounts to 0.9%. This difference in results stresses the independent nature of the tagging task in PCFG application. Tagging is viewed not as a subtask of full parsing, but as a separate problem and represents a challenge of a different kind for a PCFG. What is usually assumed as the input for parsing – a sequence of part-of-speech tags – is to be found in the tagging problem, and the main emphasis is put on identification of those regularities of syntactic structure that are highly predictive for the assignment of the correct tag sequence.

LoPar experiments

Training, tune and test sets in LoPar experiments have been compiled of Tüba-D/Z data for the same manually annotated tokens which were used in the TnT experiments.

The initial grammar and lexicon for LoPar has been extracted from the training data: the rules have been read off the parsed sentences and each rule has been associated with a number of occurrences of this rule in the training data. To accommodate the data to the experiments with LoPar, the modifications discussed below have been introduced.

The Tüba-D/Z segmentation units may contain more than one sentence and/or phrase, which results in disconnected trees in some cases. To avoid disconnected trees in the LoPar grammar files and input, a root node `VROOT` is added to all treebank segmentation units.

In the treebank, punctuation marks are not attached to trees. However, punctuation often provides useful information for correct morpho-syntactic annotation. For example, assignment of a subcategory *relative* or *demonstrative* to pronouns crucially relies on preceding tags: relative pronouns are usually preceded by a comma. To supply the tagger with this information, the punctuation is included into the rules in LoPar grammar. A sentence-internal punctuation mark is attached to the nearest node that spans tokens in both right and left context of the mark. A sentence-final punctuation mark is attached to the root node.

The tree in Figure 6.1 exemplifies the type of structures and categories used in the baseline LoPar experiment.[12] The tree structure represents the label bracketing of the original TüBa-D/Z treebank format but leaves out function argument information represented as edge labels in the TüBa-D/Z treebank.

The baseline performance of 66.40% is obtained by training LoPar on the treebank data as is.[13] Adding a back-up lexicon with candidate morpho-syntactic analyses for unknown words improves accuracy to 77.65%. Further improvements are achieved by redefinition of non-terminal nodes and tree

[12]The sentence-final punctuation is left out for expository purposes.

[13]The initial experiments are performed on the data set 1 (50k). Final results are summarized for all data sets in the evaluation subsection below.

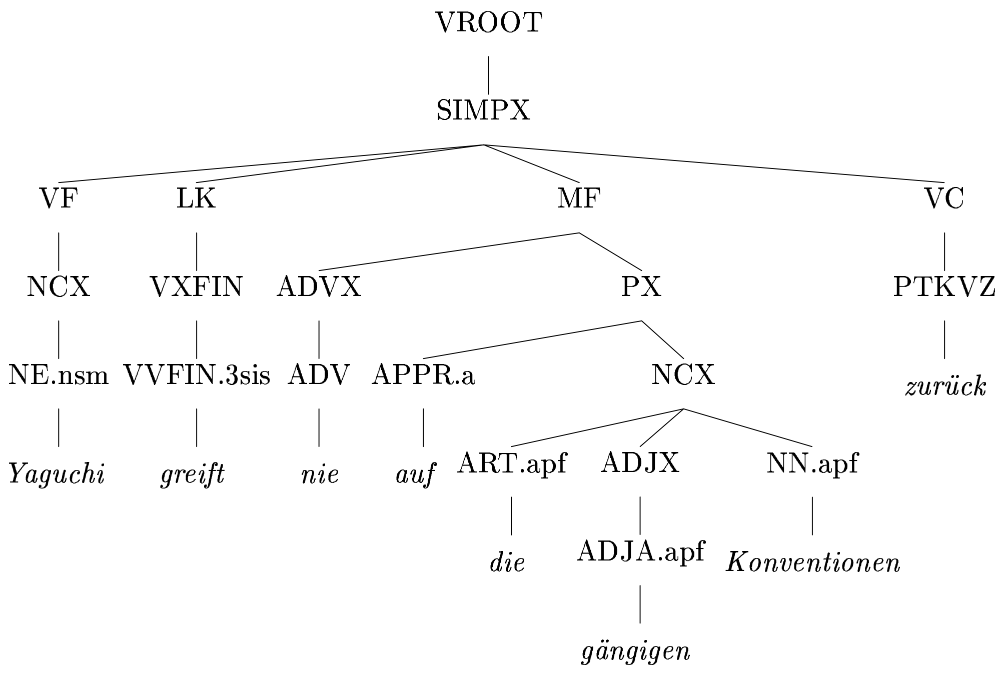


Figure 6.1: Example of an original tree

transformations compared to the original treebank. This confirms the findings of Johnson (1998) who showed that the performance of PCFG parsing varies considerably depending on the choice of non-terminal symbols and the tree transformations performed on the Penn treebank.

The transformations performed on the treebank data are described below.

Treebank transformations

Enriching the set of non-terminal nodes and tree transformations both have the goal of optimizing the treebank structure for training a model that best predicts the correct morpho-syntactic tag sequence.

Enriching the set of non-terminal nodes aims at weakening the independence assumption inherent in PCFGs. The percolation of relevant morphological and functional information between lexical and phrasal nodes allows the model to better capture regularities in syntactic structure for correct assignment of tags:

1. Passing morphological information (case, number and gender) on phrasal categories that constitute an NP (nouns, adjectives, determiners

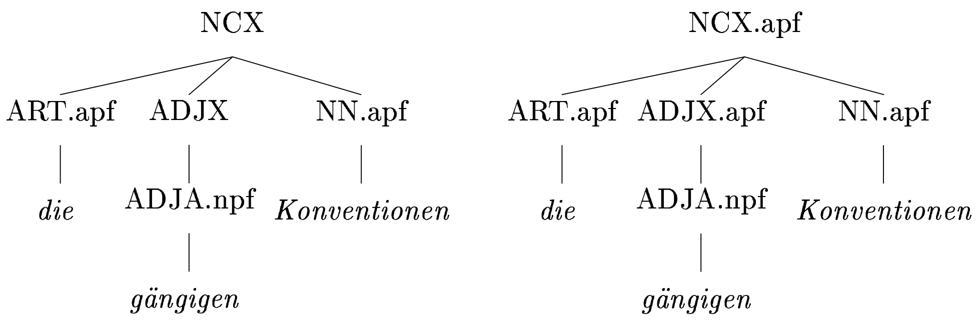


Figure 6.2: NP subtree without and with percolation of morphological features

etc.) facilitates the recognition of intra-phrasal agreement.

Consider as an example the tree in Figure 6.2. If no morphological information is encoded on the node `ADJX`, the formation of a subtree with an accusative article and a noun and a nominative adjective would be possible, since expansion of the adjectival phrase is independent of its sisters. Percolating morphological information from the adjective to its mother node makes the structure more rigid: the probability of a noun phrase expanding to categories with different morphological values is very low, as well as the probability of expanding an accusative `ADJX` to a non-accusative adjective.

2. The percolation of function labels -OA and -ON (accusative object and subject) up to SIMPX (clause) nodes helps to prevent the formation of two-subject as well as subject-less clauses, which are less frequent in the treebank.[14] In a way this information reflects the subcategorization frame of the verb. It is interesting to note, though, that inclusion of dative and genitive objects in the treebank structure hurts tagger performance.

3. Passing the FIN-label of finite verbs up to SIMPX nodes allows for the tracing of regularities between the finiteness of a clause and the morphological cases of NPs in a clause: a finite verb in a clause requires the presence of the subject, whereas the absence of a finite verb is more probable for a subject-less clauses.

4. Introducing a feature for number on -FIN and -ON nodes makes the capturing of subject-verb agreement possible.

Both -ON and -FIN labels are passed to SIMPX nodes. Trained on a treebank with correct structures, the model will consider a clause where the subject and the finite verb have different number less probable than a clause

[14]These function labels are present in the TüBa-D/Z treebank and are encoded there as edge labels between tree nodes.

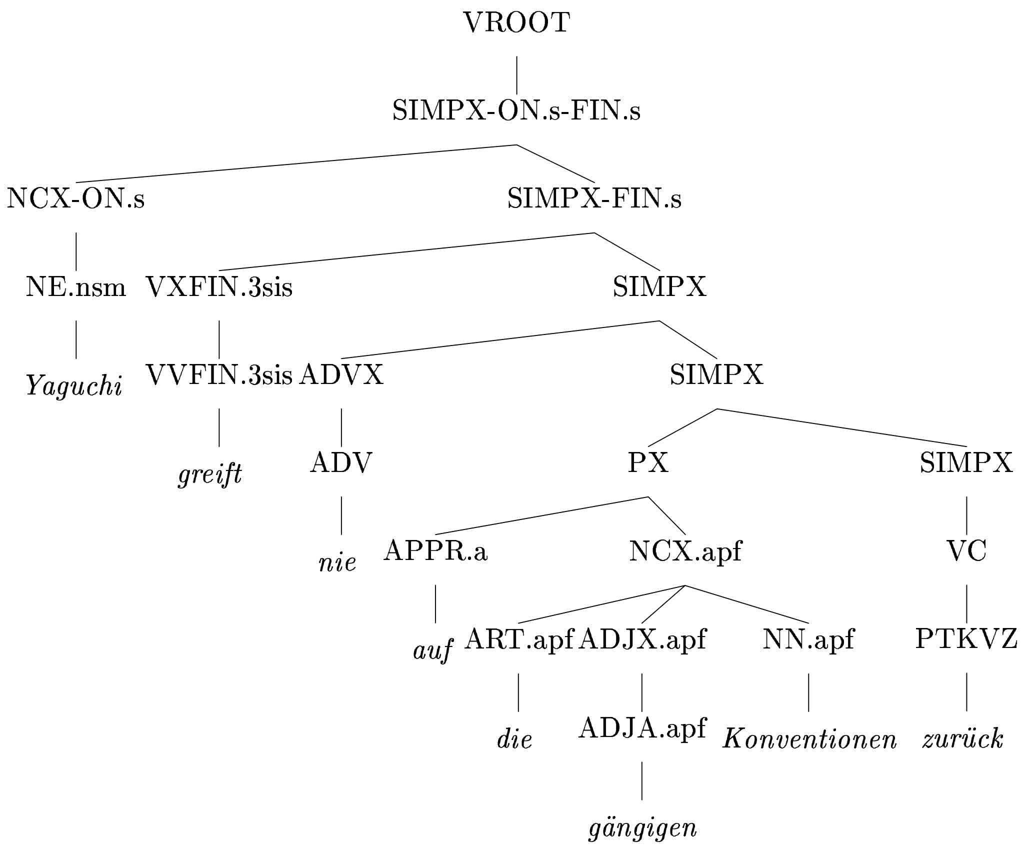


Figure 6.3: Example of a transformed tree

in which the subject and the finite verb agree in number.

The **transformation of trees** aims at two goals: to facilitate the tracing of the relevant information in the sentence structure and to overcome the data-sparseness problem created by encoding detailed information on the nodes.

1. Topological fields such as `VF`, `MF`, `NF` are present in the original treebank as an intermediate layer of syntactic structure. The elimination of all topological field information except `C` and `VC` proved advantageous since retention of only these two fields turned out to be sufficient for determining the syntactic macrostucture across different clause types. The `C` field (short for "complementizer field") and the -FIN marking on the verbal complex node reliably identify (verb final) subordinate clauses. By contrast, verb-first and verb-second clauses can be identified by the position of the finite

		preci-sion	# of unparsed sentences
1.	baseline:	66.40	5
2.	back-up lexicon added	77.65	5
3.	topological fields (except for VC and C) deleted	77.58	5
4.	case passed up to NX and NCX	84.23	6
5.	grammatical functions (-ON and -OA) added and passed up to SIMPX + rules binarized	84.98	5
6.	morph. info passed up to NXs and VXFINs	87.62	7
7.	FIN label with number passed up to SIMPX	**88.21**	9
8.	results on test data	**87.69**	11

Table 6.10: LoPar PCFG experiments

verb (without retention of the `LK` topological field) together with the absence of a -FIN specification on the verbal complex.

2. Encoding morphological and functional information on the nodes leads to a severe increase in the size of the rule set with an accompanying drop in recall (i.e. an increase in the number of unparsed sentences). This can be counterbalanced by the binarization of tree structures, which makes the grammar more flexible by allowing for structures not present in the training data to be created in the tagging phase.

Changes to the treebank were introduced in the order in which they are presented in Table 6.10. The tree in Figure 6.3 exemplifies the type of structures and categories used in the transformed treebank.

Evaluation

Table 6.10 summarizes the LoPar experiments on the tune set and the final result on the test data set 1. The baseline of 66.40% is obtained by training LoPar on the treebank data in the original format. The first improvement to 77.65% is received by adding a back-up lexicon with candidate morpho-syntactic analyses for unknown words. Further improvements are obtained by enriching the set of non-terminal nodes and by tree transformations. These improvements are summarized in lines 3 to 7 of Table 6.10.

Table 6.11 provides an evaluation of LoPar performance trained on data sets of different size. *Precision, recall* and *f-measure* are calculated in the same way as for the rule-based module. The additional metric `no tag` rep-

data	precision	recall	F-measure	no tag	LE	DE
50k	87.69%	84.50%	86.07%	3.41%	9.15%	90.85%
100k	89.05%	87.78%	88.41%	1.38%	12.72%	87.28%
150k	89.48%	88.52%	89.00%	1.08%	6.44%	93.56%

Table 6.11: Evaluation of the PCFG model

data	POS	case	number	gender	person	tense	mood
50k	34.56%	38.35%	1.11%	7.21%	1.76%	1.20%	0.46%
100k	36.70%	37.77%	1.55%	5.14%	0.65%	0.24%	0.49%
150k	28.79%	43.02%	1.81%	5.70%	1.28%	0.34%	1.41%

Table 6.12: Error analysis for the PCFG model

resents the percentage of tokens for which no tag was assigned. The tagger is unable to assign a tag to a token if the PCFG cannot provide any parse for a sentence due to data sparseness.[15]

In case of the PCFG model, errors are considered lexical if the correct tag is not present in the lexicon of the tagger. Otherwise they are viewed as disambiguation errors.[16] Table 6.12 shows the error distribution among the morpho-syntactic features for the PCFG model.

As compared to the performance of the TnT model without the loss of information in the underlying tagset, the performance of the PCFG model represents an improvement of 2–5%, which amounts to a significant error rate reduction: 15.97–36.51%. It is also worth noting that the size of training corpus required to achieve acceptable results for the PCFG model is rather modest (50k). The fact that the model can be trained successfully on manually annotated treebanks of such modest size highlights the feasibility and applicability of the overall approach.

Discussion

Some of the changes introduced to the treebank data improve performance only in conjunction with others, such as the cluster of transformations described in line 5 of Table 6.10. These transformations include adding and percolating grammatical functions in conjunction with rule binarization.

[15]This adverse effect on recall is clearly due to the modest size of the training data and should be alleviated as more training data are added.

[16]Distribution of errors is given only for tagged tokens.

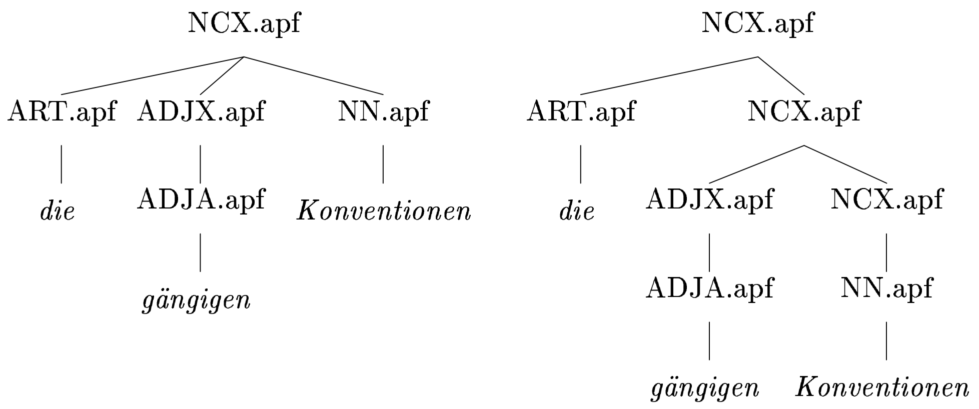


Figure 6.4: An original and a binarized structure of an NP

When introduced independently of the others, any of these three changes turn out to deteriorate performance. The introduction of grammatical functions on the nodes aims at capturing clausal structure regularities. If the grammatical function information is not passed to parent nodes, the capturing of regularities is complicated. Therefore, the introduction of the information without passing it to parent nodes leads to the effect opposite to the desired. The percolation of the information, however, results in a significant expansion of the rule set. To alleviate the arising data sparseness problem, binarization of the rules is required. The union of the three changes yields an improvement in performance and reduces the number of unparsed sentences.

The binarization of tree structure decreases performance unless it is accompanied by the expansion of the rule set. The reason for this lies in the nature of the operation: it is oriented onto reducing the data sparseness problem and is not based on linguistic intuitions. Often this transformation leads to the creation of ungrammatical syntactic structures. Consider, for example, an original and a binarized structure of the same noun phrase presented in (6.4). Table 6.13 lists rules used in the derivation of the binarized structure. This set of rules authorizes the ungrammatical structuring of NPs with more than one article.

To avoid such problems, only clausal nodes but not phrasal nodes are binarized in the experiments with LoPar. This is justified from the point of view of how enriching node labels with grammatical function and morphological information influences the size of the rule sets for clausal and phrasal categories. The number of rules used in the derivation of the phrases in the treebank does not increase significantly when morphological informa-

NCX.apf → ART.apf NCX.apf
NCX.apf → ADJX.apf NCX.apf
ADJX.apf → ADJA.apf
NCX.apf → NN.apf

Table 6.13: Rules used in the derivation of the binarized structure in Figure 6.4

tion is introduced to the nodes. This is due to the fact that daughters in a phrase agree in their morphological characteristics, and therefore, the variety of possible morphological values for the nodes in a phrase is restricted to those that are incorporated on other nodes in the phrase. Consider, for example, an original NP structure in Figure 6.4. Given noiseless training data, the rule "`NCX -> ART.apf ADJX NN.apf`" will have only one equivalent, "`NCX.apf -> ART.apf ADJX.apf NN.apf`", after enriching NCX and ADJX nodes with morphological information.

Clausal nodes are different in this regard. Consider a rule "`SIMPX -> NCX VVFIN NCX`" which sanctions the derivation of many clauses with different case values of noun phrases. Some of such clauses are presented in (59)–(63):

(59) Er ist mein Bruder.
he_{nom} is my_{nom} brother_{nom}

(60) Mein Bruder hat viele Kinder.
my_{nom} brother_{nom} has many_{acc} children_{acc}

(61) Viele Kinder hat mein Bruder.
many_{acc} children_{acc} has my_{nom} brother_{nom}

(62) Mein Bruder hilft mir.
my_{nom} brother_{nom} helps me_{dat}

(63) Mir hilft mein Bruder.
me_{dat} helps my_{nom} brother_{nom}

The enrichment of the NCX nodes with morphological and grammatical function information will result in several rules instead of one. Therefore, binarization of clausal nodes is necessary.

Apart from the transformations described above, other transformations on the treebank data have been tested in the experiments. Error analysis has shown that the tagger often marks arguments of copula verbs as nominative and accusative objects, which suggests that the tagger has not been able to detect prototypicality of two nominative arguments for such verbs. To facilitate the capture of this regularity, a grammatical function `Pred` standing for *predicative object* and a marker `Cop` for copula verbs have been introduced to the tree structures and percolated to parent nodes. However, neither together, nor separately, did these transformations manage to improve performance. This could be due to the fact that such constructions rarely occur in the data as compared to the constructions with two arguments bearing different case value. Other experiments have shown that introduction and percolation of information which is comparatively rare in the data, such as grammatical functions *dative* and *genitive objects* tend to lead to a decrease in performance.

No improvement has been obtained by marking non-finite verbs in the same manner as finite verbs, as well as by percolating morphological information on VC nodes and passing VC marker to parent nodes. Apparently, this indicates that the morpho-syntactic values of tokens are rather independent of the information contained in the verbal complex of a clause.

6.3 Comparison of the rule-based and statistical models

The experiments described in the previous sections have demonstrated the successful application of rule-based and statistical taggers to the task of morpho-syntactic annotation of German.

The rule-based tagger significantly reduces the number of analyses of tokens and provides a high accuracy of annotation: 70% of all tokens receive a single analysis and in 97.62% of cases the analysis is correct. The statistical models provide full disambiguation of tokens. The PCFG model achieves precision that is 4.8% higher than the best result reported in the literature for the morpho-syntactic tagging of German, which corresponds to a substantial error rate reduction of 31.37%.

Comparison of the error analysis of the taggers suggests that the rule-based model performs better on morphological disambiguation: the percentage of errors made by it in morphological features is rather small as compared to the same statistics for the statistical model.

The main difference in the performance of the rule-based and the statistical models concerns, however, the ambiguity of tokens after model application. While the statistical models resolve the ambiguity completely, the rule-based model allows for multiple analyses in the output. This suggests different application of the models in tasks which require morphological disambiguation. If a unique analysis for every word in the input is required, then the PCFG or TnT model will be more appropriate. However, the more cautious rule-based model will be preferable if partially ambiguous input is acceptable which can be further disambiguated by subsequent processing modules.

The powers of the rule-based and the statistical models can be combined in a uniform hybrid tagging system in line with ideas of Hajič et al. (2001). Design and implementation of a hybrid tagging system with a rule-based and a PCFG components is described below.

6.4 Combined model

6.4.1 Architecture

The model has a layered and sequential architecture consisting of morphological analysis, rule-based disambiguation and statistical tagging. The order of these modules reflects the relative strengths of the rule-based and statistical methods involved.

The morphological analyzer provides all of the possible analyses for a given sequence of tokens. This highly ambiguous output is then fed to the rule-based module. Its task is to reduce the number of candidate analyses to be considered by the statistical module. If used cautiously, the rule-based method will rule out only those candidates for which it has sufficient evidence and will retain all those that are contextually plausible. The task of the statistical module is to disambiguate the remaining cases of ambiguity. Statistical disambiguation is made considerably easier by the rule-based pre-filtering module, since the remaining set of hypotheses is greatly reduced. This reduction in search space corresponds to a gain in precision compared to purely statistical disambiguation.

6.4.2 Implementation

Two experiments have been performed with the combined model. In the first experiment all analyses left after application of the rule-based module are provided as input to the statistical module. The search space of the statistical module is thus restricted to the readings that the rule-based component considers grammatical. In the case where there is a single analysis for a token available, it remains unchanged. The main advantage of such a strategy is the elimination of ungrammatical readings prior to probabilistic processing. This helps to avoid errors made in cases that are traditionally hard for statistical models and are comparatively easy for rule-based approaches (such as long distance dependencies among tags). The drawback of the strategy, however, consists in carrying all the errors of the rule-based component into the final model performance, which mainly concerns unknown words: the rule-based module will produce an error if the lexicon or the guesser of the morphological analyzer provides a set of hypotheses that does not include the correct analysis.

In the second experiment the input to the statistical model was limited to the categories that are most reliably tagged by the rule-based module. This provides the intended division of labor between the two modules according to their strengths: the rule-based component eliminates analyses in sure cases and performs disambiguation based on long-distance relations, whereas the statistical module resolves the remaining ambiguity and assigns tags to unknown tokens.

The input to the statistical model in the second experiment was prepared in the following way: all unambiguous analyses produced by the rule-based module are included in the input, except for the analyses of predicative adjectives (the morphological analyzer often mistakens them for adverbs), of imperative verbs (these are often erroneous analyses that are actually foreign material), and of proper nouns without morphology. In addition, all analyses that have unambiguous POS readings of article, attributive adjective, finite verb, demonstrative, relative or personal pronoun are included in the input for the statistical module, since these parts of speech in most cases are tagged correctly by the rule-based module and since resolution of remaining morphological ambiguity for them does not usually constitute a problem for the statistical module. Due to the unreliable treatment of unknown words by the rule-based component, the statistical module is used as its own preprocessor to identify the categories that most often correspond to unknown words. These categories include foreign material (FM), special symbols (XY) and pronominal adverbs (PROP). The tags for these categories are then included into the input to the combined model, replacing the analyses of the rule-based module.

In both experiments, after application of the statistical module, tokens of unparsed sentences are provided with (possibly ambiguous) analyses assigned by the rule-based module.

data	input	precision	recall	F-measure	no tag	LE	RBE	SE
50k	full	90.05%	80.68%	85.11%	10.40%	0%	32.33%	67.67%
	partial	89.49%	81.58%	85.35%	8.84%	6.88%	16.50%	76.62%
100k	full	90.13%	84.75%	87.36%	6.03%	0%	42.37%	57.63%
	partial	90.47%	86.52%	88.45%	4.45%	8.99%	17.87%	73.14%
150k	full	89.53%	86.93%	88.21%	2.90%	0%	45.76%	54.24%
	partial	90.60%	88.54%	89.56%	2.27%	11.78%	18.05%	70.17%

Table 6.14: Evaluation of the combined model

6.4.3 Evaluation

Table 6.14 presents the performance of the combined model. The first line demonstrates the performance of the model that takes the full input from the rule-based module. The second line provides statistics for the model that takes a partial input of the most reliable categories from the rule-based component. To reflect the impact of the errors by the rule-based component on the error rate of the model, the disambiguation errors are split into errors of the rule-based module (RBE; column 7) and errors of the statistical module (SE; column 8). Table 6.15 demonstrates the distribution of errors among the morpho-syntactic features.

errors	POS	case	number	gender	person	tense	mood
SE	24.07%	43.33%	0.74%	9.81%	0.37%	0.18%	1.11%
RBE	43.42%	21.00%	1.78%	2.13%	0.71%	5.34%	0.00%
all	30.69%	35.69%	1.10%	7.18%	0.49%	1.95%	0.73%

Table 6.15: Error analysis for the combined model with full input

The combined model achieves an accuracy of 90.60%, beating the pure PCFG model by 1.12%. Although the final result of the model is lower than the state-of-the-art results for morpho-syntactic tagging of other inflectional languages, such as Czech, Romanian or Hungarian, it is worth noting that the amount of data provided for the model is rather small in comparison to the size of training sets in the experiments with other languages (consider, for example, the training data of 1.8M tokens used by Hajič et al. (2001) as compared to 50-150K of tokens used for the training of the current model). Moreover, the German language represents a particularly challenging problem for morpho-syntactic annotation due to its morphological characteristics, such as high ambiguity rate and paradigm-dependent case syncretism. The size of the tagset used in the experiments also exceeds the size of tagsets reported for other languages. The fact that the model performs successfully when trained on restricted amount of German data highlights the feasibility and applicability of the overall approach.

6.5 Conclusion

This chapter has described the application of two different methods, a rule-based method and a statistical method, to the morpho-syntactic annotation of German. The methods have been chosen that provide the optimal trade-off between the amount of pre-tagged data required for implementation of a method-based model and the accuracy of annotation yielded by the model.

The implementation of two models based on these methods has been presented and the performance of the models has been evaluated. It has been shown that the rule-based model designed as a morpho-syntactic disambiguation module of the Xerox Incremental Deep Parsing System achieves a high accuracy of annotation and successfully reduces data ambiguity.

Experiments with statistical models have demonstrated the limitations of n-gram taggers to the task and a suitable alternative has been presented as a PCFG-based tagger. It has been shown how the performance of the PCFG model can be drastically improved by systematic transformations of the training data. The PCFG model has been successfully trained on a small amount of data and achieved results which outperform the current state-of-the-art results for the morpho-syntactic tagging of German.

Finally, a combined model with a rule-based and a PCFG modules has been presented. The model profits from the strengths of the methods involved and demonstrates performance exceeding the performance of the modules used independently.

Chapter 7

Application in dependency parsing

7.1 Motivation

The current chapter presents a dependency parser of the GeRman Incremental Parsing system (GRIP), a major project on the analysis of German.

One of the main distinctions of the GRIP parser from other German parsers described in the literature concerns the kinds of information which provide a basis for the analysis. Most standard dependency parsing models rely in a significant way on subcategorization information, such as verbal subcategorization frames, in the parsing process. Thus, the three German parsers described in Chapter 3, i.e. Topological Dependency Grammar, Weighted Constraint Dependency Grammar and Concurrent Lexicalized Dependency Parser, incorporate such information in the lexical entries of tokens and employ valency constraints to ensure the correct assignment of arguments to verbs. Subcategorization information significantly simplifies the parsing task, since necessary clues about obligatory and possible dependency relations of tokens are provided to the parser. However, such information needs to be collected and included in the parser lexicon, which is time and labor consuming.

In the GRIP parser, on the other hand, such complex external knowledge is reduced to a minimum and the main information source on which the parsing process is based is represented by the morpho-syntactic characteristics of tokens and by the linear order of tokens in a sentence. Development of such a parser aims at demonstrating that high parsing performance does not necessarily involve the incorporation of a rich lexicon into the grammar and

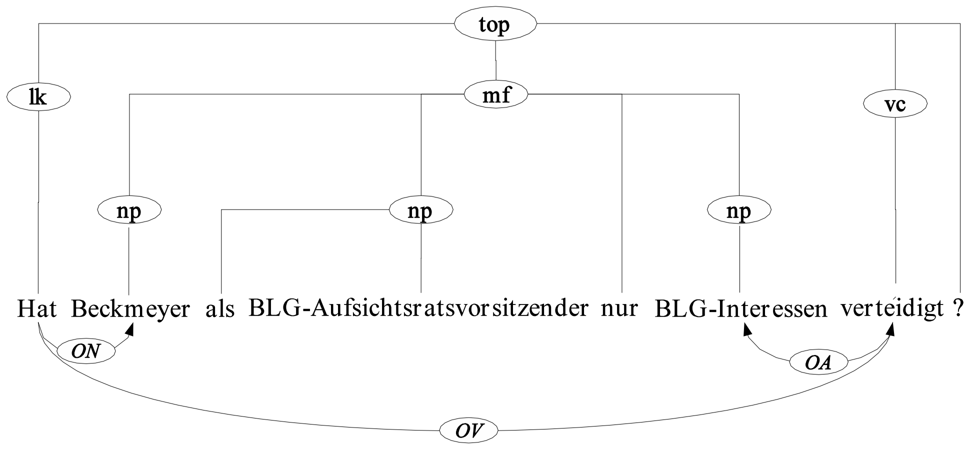

top
lk
mf
vc
np
np
np
Hat Beckmeyer als BLG-Aufsichtsratsvorsitzender nur BLG-Interessen verteidigt ?
ON
OA
OV

Figure 7.1: Analysis provided by the GRIP parser

that state-of-the-art results can be achieved if a reliable morpho-syntactic tagger is used as a pre-processor of the parser.

7.2 Introduction

The GRIP parser is a robust deterministic parser implemented in the XIP system described in Chapter 4 above. The parser is a part of the GRIP system and is composed of two modules – a chunker, which provides shallow constituency analysis for German sentences, and a dependency module, which establishes dependency relations between tokens in the input.

The ultimate goal of the parser is the assignment of dependency structures to German sentences. In the current version, the parser concentrates on the annotation of the frame of a sentence: the parser identifies the main element of the sentence and its arguments, i.e. a verbal group and its complements. Figure 7.1 exemplifies kind of analysis provided by the parser.

Constituency analysis plays a supporting role in the dependency analysis. By grouping lexical tokens in phrases, it pre-defines the possible domains of dependencies, which significantly simplifies the process of dependency assignment. Thus, for example, a direct object relation can possibly be established between any verb and any noun in accusative case. With a preprocessing constituency analysis which identifies phrases and topological fields, the search space is easily restricted to the heads of nominal phrases in the initial and the middle fields relevant for the verb. This excludes from consideration all nouns in other clauses, nouns occurring in prepositional and adjectival phrases and non-head nouns in nominal phrases.

The general annotation scheme provided by the parser follows the principles adopted for annotation of the Tüba-D/Z treebank. Minor modifications introduced to the treebank scheme are discussed in the relevant sections.

7.3 Constituency analysis

7.3.1 Constituents inventory of GRIP

The output structures of the chunker are based on the TüBa-D/Z treebank structures. However, the purpose of the chunker (providing a basis for the dependency module) has determined systematic changes to the original treebank style of annotation. The changes have lead to a more shallow annotation.

Table 7.1 lists phrasal and topological field constituents annotated by the chunker. Both the GRIP and the corresponding Tüba-D/Z constituency labels are given, as well as a short description of the constituency.

GRIP	**Tüba-D/Z**	**description**
NP	NCX	non-recursive noun phrase
PP	PX	prepositional phrase
AP	ADJX	adjectival phrase
VF	VF	initial field
LK	LK	left sentence bracket
MF	MF	middle field
VC	VC	verb complex
NF	NF	final field
CF	C	complementizer field
KOORD	KOORD	field for coordinating particles
PARORD	PARORD	field for non-coordinating particles

Table 7.1: Constituents annotated by the GRIP parser

Some of the phrasal nodes present in the treebank represent an intermediate layer of syntactic structure which makes dependency assignment more complicated. To prevent this complication, the following nodes are not annotated in the chunking module:

- phrasal nodes: ADVX, FX, VXFIN, VXINF;
- coordinated fields: FKONJ, FKOORD;
- others: DM, DP, EN, EN-ADD, LV.

Moreover, the chunker is not oriented to the annotation of deep structures and leaves this task for the dependency module of the grammar. Relevant nodes of the treebank which include NX, R-SIMPX, SIMPX are ignored at this stage. Furthermore, the treebank style does not make a distinction between prepositional chunks and prepositional phrases (both are annotated as PX). The chunker annotates only prepositional chunks, i.e. it leaves out annotation of recursive prepositional phrases.

A virtual root node TOP is added to every GRIP analysis automatically. Other auxiliary nodes have been introduced in the chunker analysis to simplify the process of annotation. Such nodes are used for the combination of tokens under the same part-of-speech category, such as the grouping of tokens "*jede, (, r,)*" under a single node "*jede(r)*" with a label PRON, or the grouping of adjacent cardinal tokens under the same node with a label CARD, which is useful for the identification of telephone numbers. The annotation of auxiliary nodes also serves for the correction of a part-of-speech tag assigned in the input, as it is done, for example, in the case of a shortened form "*'ne*" of the article "*eine*", which receives a wrong analysis. The auxiliary nodes include ADJ, ADV, CARD, DET, FM, NOUN, PRON, VERB and a node for a subordinate clause SCL.

Figure 7.2 illustrates a tree annotated in the chunker style as compared to the treebank annotation style in Figure 7.3.[1]

7.3.2 GRIP chunking rules

GRIP chunking rules represent constraints on part-of-speech categories of the tokens, on limited lexical information and on a linear order of tokens in the input string. The rules identify a set of nodes to be combined under the same mother node and specify a context in which the creation of such new structure is valid. The context can extend as far as sentence boundaries but in practice is usually limited to a small window of adjacent tokens. The application of rules is deterministic: once a node has been created, it is not reconsidered on the later stages of analysis.

The chunking grammar consists of the following components:

[1]For expository purposes, the tree in Figure 7.3 leaves out argument function information which is annotated in the treebank on edge labels.

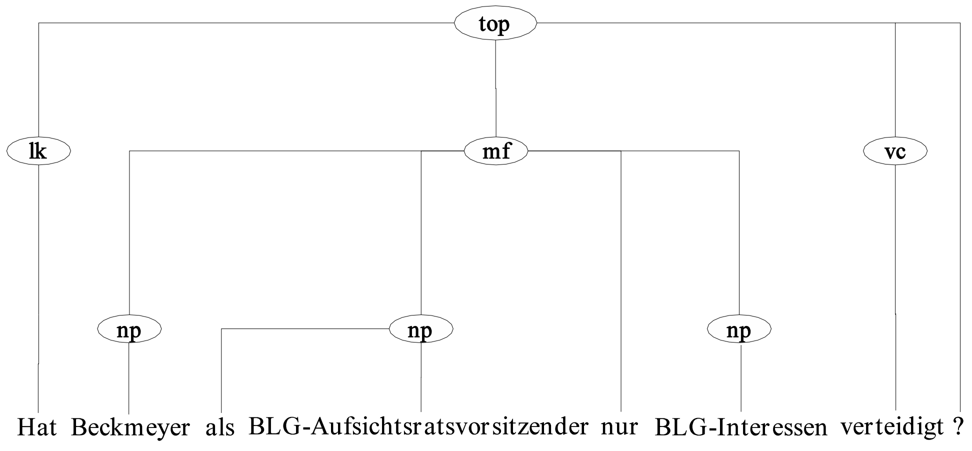

top
lk
mf
vc
np
np
np
Hat Beckmeyer als BLG-Aufsichtsratsvorsitzender nur BLG-Interessen verteidigt ?

Figure 7.2: A tree annotated in GRIP

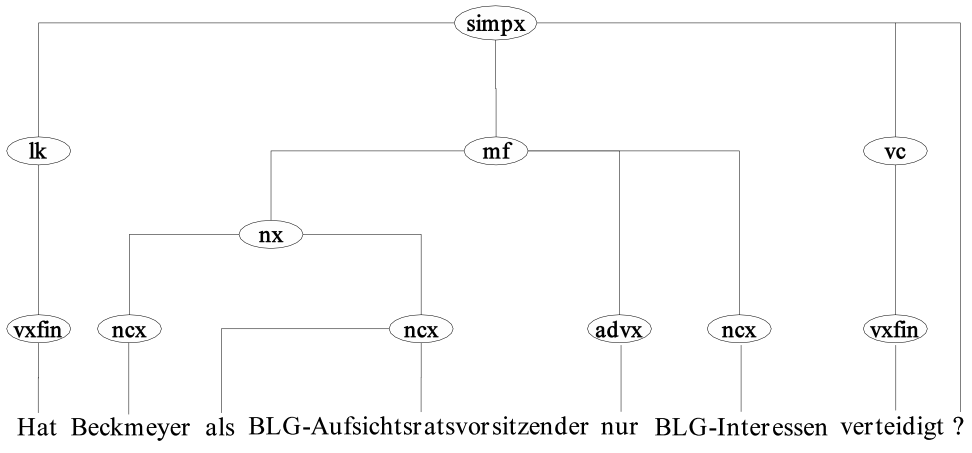

simpx
lk
mf
vc
nx
vxfin
ncx
ncx
advx
ncx
vxfin
Hat Beckmeyer als BLG-Aufsichtsratsvorsitzender nur BLG-Interessen verteidigt ?

Figure 7.3: A Tüba-D/Z counterpart of the GRIP tree in Figure 7.2

- a preprocessing component,
- a component for annotation of phrasal nodes, and
- a component for annotation of topological fields.

The preprocessing component is used for the correction of POS categories of input tokens, for grouping tokens under a common mother node with an appropriate POS category and for the preliminary structuring of set phrases into phrasal nodes. An example rule (64) of the preprocessing component is designed for identification of nouns which include a truncated part in parentheses, such as "*(Musik-)Geschichte*" ("*history (of music)*"). Using features *first* and *last*, the rule specifies the first and the last elements of the node sequence to which it applies. Additionally, a linear order of intermediate elements is determined, which restricts the area of rule application to sequences in which a truncated element precedes a right parenthesis. If the constraints on the order of tokens are satisfied, the tokens are combined under a common mother node with the category `NOUN`.

(64) noun -> punct[lpar,first], trunc#1, punct#2[rpar], noun[last], where (#1<#2).

The proper chunking is performed by the component for annotation of phrasal nodes. The component consists of constraint rules for the annotation of adjectival phrases, noun phrases and prepositional phrases. Apart from annotation of simple phrases consisting of standard elements, such as a prepositional phrase "*in einer internen Kontrolle*" ("*in an internal control*"), the chunker provides annotation of recursive phrases, such as a phrase presented in example (65):

(65)

in	der	am	vergangenen	Montag	abgesegneten	rot-grünen
'in	the	on	last	Monday	approved	red-green

Neufassung
new version"

in the red-green version approved last Monday

The annotation of recursive phrases is ensured by the repetitive statement of rules.

Additionally, complex adjectival phrases with prepositional phrase modification, such as the phrase "*rechtzeitig zum Muttertag*" ("*in time for the Mother's Day*"), are annotated by the chunker if the context provides enough evidence for an unambiguous analysis. Thus, since only one phrase can occur in the initial field position, a complex adjectival phrase with prepositional phrase modification is annotated in the example sentence (66):

(66) Rechtzeitig zum Muttertag kommen am Wochenende
'in time for the Mother's Day come on the weekend
Rosen mit dem neuen Label "Aus menschen- und
roses with the new label "from people- and
umweltschonender Produktion" in deutsche Blumenläden.
environment caring production" into German flower shops'
In time for the Mother's Day, roses with the new label "From production which takes care about people and environment" come to German flower shops during the weekend.

Example (67) provides rules that ensure the correct annotation of complex adjectival phrases with prepositional phrase modification in the initial field position. The rules refer to the node sequence of an adjectival phrase, a prepositional phrase and optional adverbs and commas. These nodes are combined together under an `AP` label if they are followed by a finite verb or by a sentence-final punctuation and if they are preceded by a sentence-final punctuation, by a hyphen or by a sentence-initial quotation mark. Another possibility for the left context represents beginning of a sentence.

(67) AP -> || adv*, AP, PP, punct*[comma] |(punct[skip]), verb[fin]; punct[sent]|.
AP -> |punct[skip,start]; punct[sent]; punct[spec]| adv*, AP, PP, punct*[comma] |(punct[skip]), verb[fin]; punct[sent]|.

After the application of the phrasal nodes annotation rules, an input string receives an analysis which includes marking of adjectival phrases, noun phrases and prepositional phrases. The following component provides further annotation of the string in terms of topological field categories. The fields are annotated in the following order: `CF, VC, LK, KOORD, PARORD, NF, VF, MF`. This order simplifies the process of annotation, since identification of consequent fields can rely on previously annotated categories. For example, in complicated cases, the correct assignment of verbal complexes requires reference to a complementizer field. The annotation of the left bracket of a sentence (`LK`) is considerably simplified if verbal complexes have already been recognized: in this case, all finite verbs which have not been assigned a `VC` mother node receive a left bracket analysis.

After the first round of the annotation of topological fields, subordinate clauses `SCL` are recognized as sequences of topological fields. The linear order of topological fields in a clause is fixed. This characteristic makes sequence rules of the XIP system particularly well suited for the task of clause annotation, since the rules explicitly identify a linear order of elements to be grouped together. Example (68) illustrates a kind of sequence rules for the annotation of subordinate clauses. The rule is designed for the annotation of subordinate clauses with the coordination of verbal complexes.

(68) SCL @= CF, (MF), (conj[neben]), (CF), (MF), (punct[skip]), VC, punct[comma], (MF), VC, punct[comma], (MF), VC, conj[neben], (adv[fauch]), (MF), VC, (punct[comma]), NF.

The annotation of subordinate clauses allows for the easier recognition of recursive topological fields, such as final fields which are represented by a clause, or middle fields with an embedded clause. The repetitive application of rules guarantees the correct annotation of recursive topological fields.

In total, GRIP constituency module comprises 1328 rules, which include 77 preprocessing rules, 464 rules for the annotation of phrasal nodes and 787 rules for the annotation of topological fields and subordinate clauses.

7.3.3 Evaluation

The performance of the GRIP chunker has been evaluated against the treebank data. A gold standard file in a bracketed format has been generated from the tree structures of the treebank. To bring the treebank and chunker structures into the same format for evaluation purposes, the following changes to the gold standard and the test file have been made:

1. Sentence-internal punctuation (i.e. `" , ( ) - '`) has been deleted from both gold and test files.

   In the treebank all the punctuation is attached to the root node. Bracketed format of the chunker output does not allow to follow this style - in the chunker output the punctuation is attached to the relevant phrasal nodes. To make the comparison possible, the sentence-internal punctuation is deleted.

2. Systematic treebank transformations have been performed:

   - Nodes that are not present either in the treebank or the chunker output format have been deleted (for the list of such non-shared nodes see discussion above);

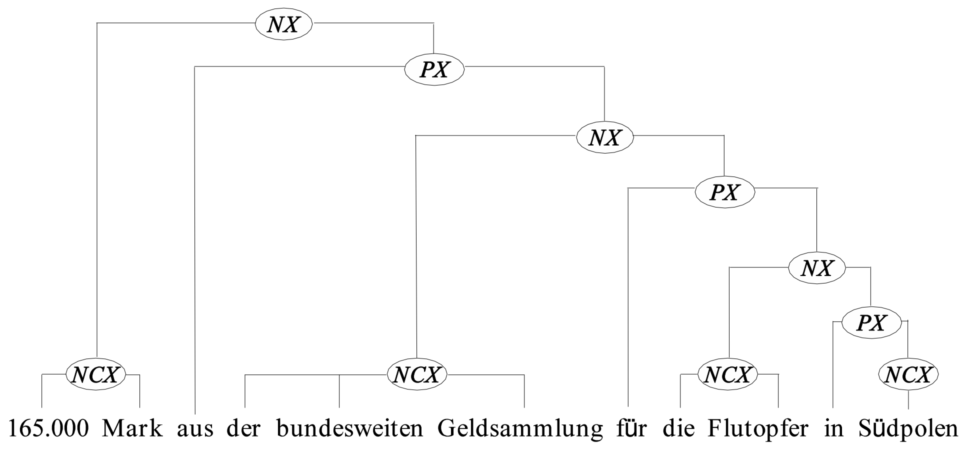

NX
PX
NX
PX
NX
PX
NCX
NCX
NCX
NCX
165.000 Mark aus der bundesweiten Geldsammlung für die Flutopfer in Südpolen

Figure 7.4: Treebank annotation with recursive prepositional phrases

- Prepositional phrases have been split to prepositional chunks (see Figures 7.4 and 7.5 for illustration).

3. Treebank categories have been renamed as shown in Table 7.2:

ADJX	->	AP
C	->	CF
ENX	->	NP
NCX	->	NP
PX	->	PP

Table 7.2: Renaming of Tüba-D/Z constituents for the evaluation

The test data used in the evaluation of the chunker comprises 12 020 tokens.[2] The average sentence length in the data set is 14.9 tokens per

[2]This test data set differs from the test data sets used in the experiments with the morpho-syntactic tagging described in Chapter 7, since part of the tagging test data has been seen during the development of the GRIP chunker. The new test data set for the evaluation of the GRIP system has therefore been compiled from the largest test data set used in the tagging experience by eliminating those sentences that have been seen during development of the GRIP system.

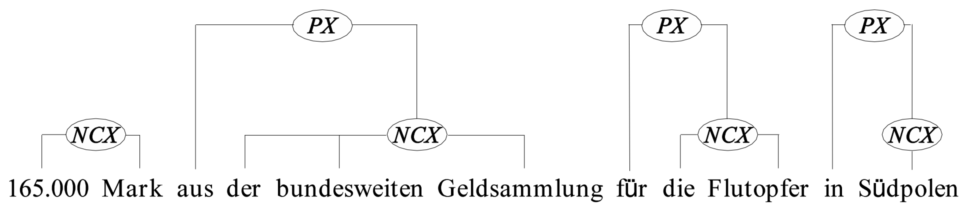

PX
PX
PX
NCX
NCX
NCX
NCX
165.000 Mark aus der bundesweiten Geldsammlung für die Flutopfer in Südpolen

Figure 7.5: Flattening of the treebank recursive prepositional phrases

sentence.

The evaluation of the chunker performance is based on the phrase boundaries. Metrics of precision, recall and f-measure, both labeled and unlabeled, have been used. For labeled metrics, not only correct spanning of a constituent is required, but also a correct labeling of the constituent.

$$(Labeled)\ Precision = \frac{\#matched\ brackets}{\#brackets\ in\ test\ data}$$

$$(Labeled)\ Recall = \frac{\#matched\ brackets}{\#brackets\ in\ gold\ data}$$

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

Table 7.3 presents results of the experiments with the chunker when part-of-speech tags are provided in the input. The table additionally provides an average number of constituents in gold and test data (first two columns, `Brackets gold` and `test`).

Brackets		**Labeled**			**Unlabeled**		
gold	**test**	**Recall**	**Prec.**	**F-meas.**	**Recall**	**Prec.**	**F-meas.**
13.9	13.6	95.31	96.43	95.87	95.71	96.78	96.24

Table 7.3: Evaluation of the GRIP chunker

7.3.4 Error analysis

The most common types of errors made by the chunker can be grouped under four categories:

- clause boundaries errors;
- coordination errors;
- errors due to complex sentence structures;
- errors due to conscious differences in annotation style.

The **first** type of errors concerns erroneous annotation of topological fields. After a complementizer field `CF` and sentence brackets `LK` and `VC` are annotated, other topological fields are recognized as sequences of nodes between the brackets. Thus, the initial field is found between the initial clause boundary and the left bracket, the middle field is limited to nodes between (a) a complementizer field or a left sentence bracket and (b) a verbal complex or a final clause boundary, and the final field usually represents a sequence of nodes after the verbal complex. However, embedded clauses considerably complicate the annotation of fields. If a sentence contains coordination, it is often difficult to state without reference to the meaning of a sentence, whether a verbal complex belongs to the embedded clause or to the main clause, as in the case of sentence (69):

(69) Schließlich kann jeder Mittelstufenschüler des örtlichen
'finally can every middle school student the$_{gen}$ local$_{gen}$
Vorstadtgymnasiums nachfühlen, daß abhängen, sein Ding
suburb$_{gen}$ gymnasium$_{gen}$ feel, that to hang out, his thing
durchziehen, kiffen, Autos, Anlagen und Computer zu
to do, to smoke pot, card, investments and computer to
besitzen und sich ab und zu mal zu verknallen
own and oneself now and then to fall in love the
die Sache mit der Pubertät ein wenig leichter macht.
matter with the puberty a little easier makes'

Finally, every middle school student can feel that hanging out, doing one's own thing, smoking pot, owning cars, investments and computers and falling in love every now and then makes the matter with the puberty a little easier.

Moreover, when several clauses follow each other in a sentence, the final field of a clause can be confused with the initial field of a following clause. A failure to identify clause boundaries leads to erroneous analysis of several fields, and ultimately results in additional wrong annotation of dependencies.

The **second** type of errors concerns coordination constructions. The errors arise if possible structural ambiguity prevents a chunker from grouping nodes correctly. Such errors are specific for phrasal nodes. An example of such an error is an error in the annotation of a prepositional phrase in example (70). The prepositional phrase spans for all the tokens in the example. However, a chunker has erroneously analyzed only first four tokens as a prepositional phrase and has not included last two NPs in the structure.

(70) mit seinen skurrilen Gestalten, grandiosen Gesichtern und
'with his bizarre shapes, terrific faces and
unbezahlbarem Witz
invaluable wit'

Errors of the **third** type arise in sentences with a complex structure and/or unusual phenomena. Texts of a newspaper style contain many sentences with heavily embedded clauses, parenthetical constructions, unexpected punctuation and other phenomena difficult for automatic processing. An example of such a complicated sentence is presented in (71):

(71) Die halbe Hundertschaft deutscher Talkmaster – doch, doch,
'The half group of hundred German$_{gen}$ Talkmaster – yes, yes,
so viele gibt es wirklich! – hatte der Diplomand von der Kölner
so many gives it really! – had the graduand from the Cologne
Kunsthochschule für Medien angeschrieben, damit sie für ihn
art college for media written down, so that they for him
und seine Abschlußarbeit vor laufender Kamera letztlich
and his final paper before working camera ultimately
nichts anderes tun als in einer gewöhnlichen Talkshow: dasitzen
nothing else do than in a usual Talkshow: sit
eben, sich räuspern, vielleicht ein Schlückchen Wein trinken
evenly, hem, perhaps a sip wine$_{gen}$ drink
usw. – nur daß sie dabei, so Wilkes' Einfall, kein Wort reden
etc. – only that they there, so Wilkes' idea, no word to say
sollten.
should.'

The graduand from the Cologne art college for media had written down half a hundred of German talkmasters – yes, yes, there are so many indeed! – so that they ultimately do nothing else for him and his final paper in front of a camera than in a usual talk-show: sit evenly, hem, perhaps drink a sip of wine etc. – only that they should not, according to Wilkes' idea, say a single word.

The **last** type of errors include errors which are due to conscious discrepancies of the annotation style of the chunker and the treebank. They mainly concern treatment of nodes that are unattached in the treebank, such as discourse markers of the kind presented in Figure 7.6. The bracketed format of the GRIP output does not allow for unattached nodes. Therefore, a discourse marker is included in the structure of the sentence. Such discrepancies between the annotation styles lead to a decreased number in the evaluation.

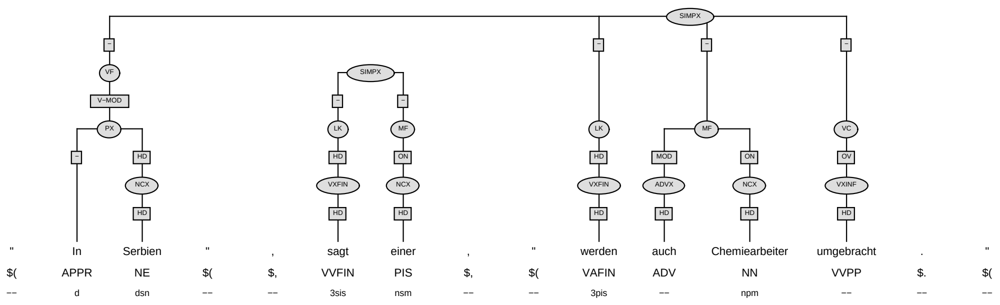

SIMPX
VF
V-MOD
PX
HD
NCX
SIMPX
LK
MF
VXFIN
ON
LK
MF
VC
MOD
ON
OV
ADVX
NCX
VXINF
" In Serbien " , sagt einer , " werden auch Chemiearbeiter umgebracht . "
$(APPR NE $($, VVFIN PIS $, $(VAFIN ADV NN VVPP $. $(
d dsn 3sis nsm 3pis npm

Figure 7.6: A TüBa-D/Z segmentation unit with a discourse marker

(72) “In Serbien”, sagt einer, “werden auch Chemiearbeiter
“in Serbia”, says somebody, “get also chemical workers
umgebracht.”
murdered”
‘“In Serbia”, somebody says, “chemical workers are also murdered”.’

7.4 Dependency assignment

7.4.1 Dependency inventory of GRIP

The dependency inventory and the principles of dependency assignment of the parser are based on the annotation scheme of the Tüba-D/Z treebank. Table 7.4 lists dependency relations currently annotated by the parser. The first column represents the name of the dependency assigned by the parser, the second column provides the corresponding name of the dependency in the treebank and the last column contains a description of the dependency.

GRIP	**Tüba-D/Z**	**description**
`SUBJ`	`ON`	subject
`DOBJ`	`OA`	direct object
`OD`	`OD`	indirect (dative) object
`OG`	`OG`	genitive object
`OS`	`OS`	sentential object
`OV`	`OV`	verbal object
`PRED`	`PRED`	predicative object
`VPT`	`VPT`	separable verbal particle

Table 7.4: Dependency relations annotated by the GRIP parser

The first three dependencies in the table correspond to the classical dependencies of subject, direct object and indirect object. Subject relation includes the expletive subject "*es*" as in (73) and sentential subject as in (74):

(73) Es regnet.
'It rains.'

(74) Daß die Ordnungsmuster ein wenig durcheinandergeraten sind, ist
That the order samples a bit mixed up are, is
heute Standard.
today standard.

The sentential object, as in (75), represents a dependency distinct from the direct object and is marked with the label `OS`:

(75) Ob die Bomben tatsächlich etwas bewirken, weiß ich
Whether the bombs actually something bring, know I
nicht.
not.
'Whether the bombs actually bring something, I don't know.'

If a dependency relation involves a sentence, as in sentential subject and object relations, the dependency is established between the main element of a sentence and its governor. Thus, in sentence (74), a relation is established between the verbs “*sind*” and “*ist*”. In sentence (75), a relation connects the verbs “*bewirken*” and “*weiß*”.

Genitive object relations marks a dependency between a verb and its object in the genitive case, such as “*eines Nachtrags bedürfen*” (“*to require an addition*”).

Verbal object dependency is established between the elements in a verbal chain, as, for example, between the verbs “*werden*” and “*soll*” and between the verbs “*gemacht*” and “*werden*” in sentence (76):

(76) Durch eine Spezialschiene soll sein Mitwirken doch noch
through a special splint should his participation however still
möglich gemacht werden.
possible made become.
'His participation should however be made possible through by the use of a special splint.’

With predicative object relations, verbs and their predicates are connected:

(77) Das Altenheim **sei** “ein **Prestigeobjekt** von ihr und
The retirement home be “a object of prestige of hers and
anderen”.
others
‘The retirement home is claimed to be an object of prestige of hers and others.’

A dependency marked as separable verbal particle is established between two parts of a separable verb:

(78) Und die Männer **machen mit**.
And the men take part.

Additionally, the GRIP analysis identifies the root of the dependency tree and marks it with label `ROOT`.

7.4.2 Decisions

In the treebank, dependency relations are encoded indirectly in a form of argument functions assigned to nodes. For example, a subject relation between the finite verb “*müssen*” and the noun “*Anbieter*” in the sentence “*Private Anbieter müssen der Telekom Schadenersatz bezahlen*” pictured in Figure 7.7 is identified as an edge label `ON` above the noun phrase node

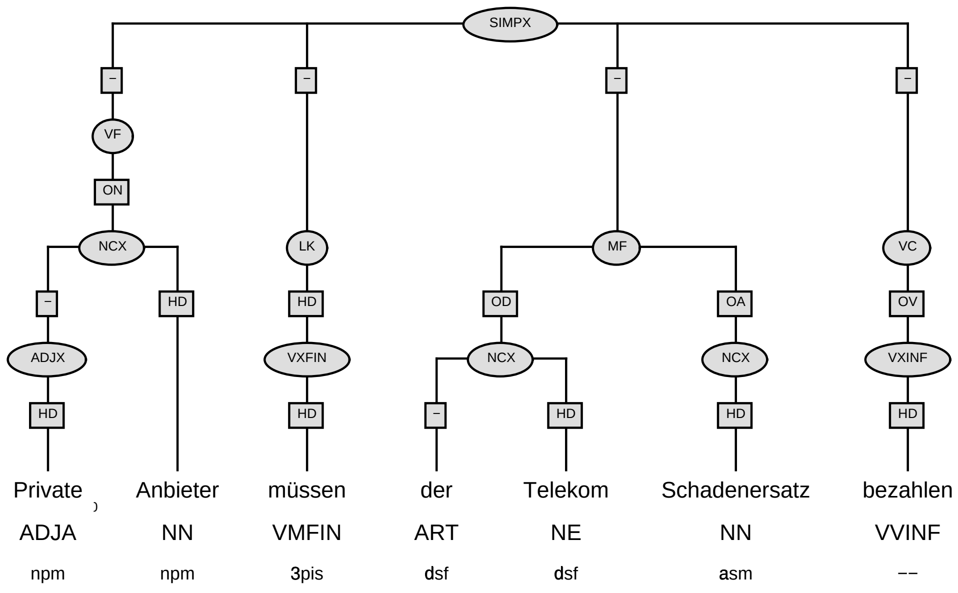


Figure 7.7: An example TüBa-D/Z tree

"*Private Anbieter*". A governor of the constituent can be found with a heuristic that searches for a head sister of the constituent and extracts the lexical head of the head sister as a governor of the constituent in question. However, in the case of non-headed constructions, such as a sentence or a coordination phrase, the treebank style of annotation leaves a choice as to the establishment of a dependency relation.

In the case of sentential nodes, the question arises about the possible attachment of verbal arguments, the alternatives being a finite verb and a main verb. According to the GRIP style of annotation, subjects and verbal objects are assigned to the finite verb of a sentence, whereas all the other verbal arguments are attached to the main verb. This decision is guided by the following reasons: since morphological relation (expressed in agreement of features) connects the finite verb and a subject, it is sensible to establish a syntactic relation between these two elements as well. However, it is the main verb rather than the auxiliary verb that subcategorizes for the nominal arguments. Therefore, all other arguments are assigned to the main verb. In general, a verbal group (a main verb together with an auxiliary) can be seen as a nucleus, since it contains both a semantic and a syntactic

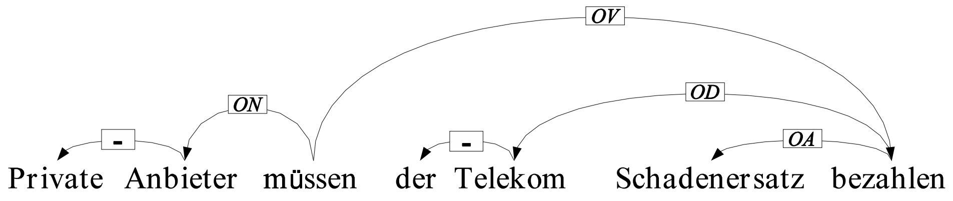


Figure 7.8: A dependency structure for the tree in Figure 7.7

center. Thus, all the arguments in the sentence, such as subject, direct and indirect objects, etc., can be treated as dependent on the verbal nucleus but attached to different parts of the nucleus in the output of the parser due to morphological reasons.

For illustration purposes, a dependency structure of the "*Private Anbieter müssen der Telekom Schadenersatz bezahlen*" is presented in Figure 7.8.

Another case in which a decision had to be made externally of the treebank annotation style concerns the treatment of coordinate structures. Since in the treebank such structures do not have a head, in this case the mapping of a constituency tree into a dependency structure is not straightforward. Following Hudson (1990), the GRIP system treats a coordinate structure as a "*word string*", i.e. as a syntactic unit without an internal head-modifier distinction. To reflect the decision in a dependency annotation, each conjunct of the structure is connected to the governor of the coordinate structure. In the same manner, a modifier of the coordinate structure holds a dependency relation to each conjunct of the construction. The conjunction is attached to the last conjunct of the construction.

Figures 7.9 and 7.10 illustrate the treatment of coordinate structures in the treebank and the GRIP system, correspondingly.

The last deviation of the GRIP annotation style from the treebank style concerns the orientation of dependency in prepositional phrases. According to the Tüba-D/Z style, the head of a prepositional phrase is a noun. The GRIP system adopts a different view and marks the preposition as the head of a phrase.

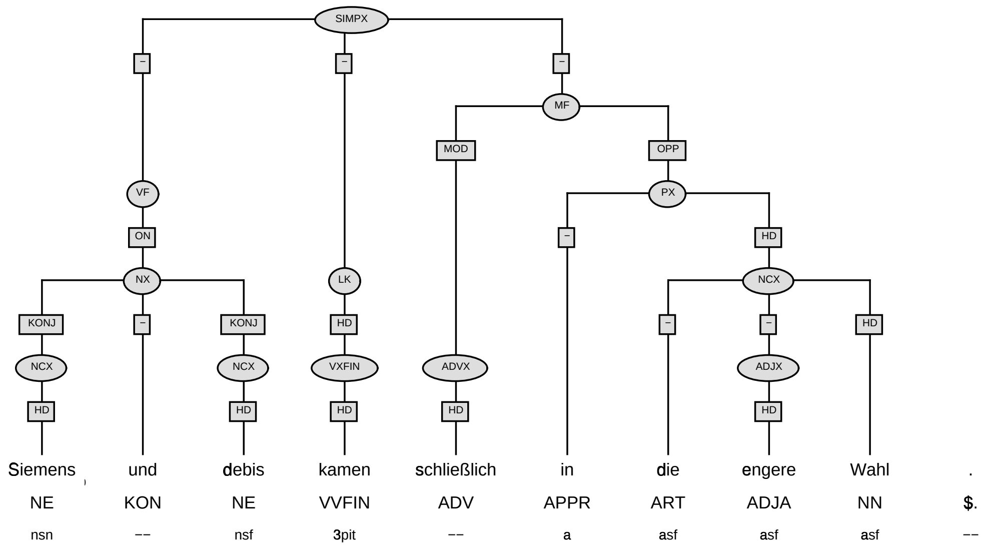


Figure 7.9: A TüBa-D/Z tree with coordination

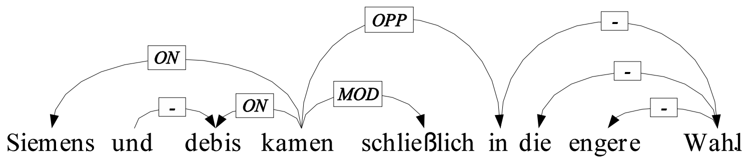


Figure 7.10: A dependency structure for the tree in Figure 7.9

7.4.3 GRIP dependency rules

The GRIP dependency module assumes a pre-chunked input in the bracketed format, as provided by the GRIP chunker. Nodes in the input are associated with relevant (possibly ambiguous) morphological and categorial information. Dependency annotation also relies on restricted information contained in the GRIP lexicon: when appropriate, the lexicon assigns one or more of the features listed in Table 7.5 to high frequency verbs:

Feature	**Example**
ditransitive	*fragen* ("to ask")
with genitive object	*bedürfen* ("to require")
reflexive	*sich bedienen* ("to help oneself")
separable	*aussehen* ("to look")
performative	*berichten* ("to report")
with predicative object	*bleiben* ("to remain")

Table 7.5: Features assigned to verbs in the GRIP lexicon

The lexicon has been compiled based on the training data and contains approximately 150 verbs.

In the GRIP system, dependency relations are annotated sequentially, so that annotation on later stages can rely on previously assigned dependencies. Apart from the set of dependency relations listed in Table 7.4 above, the auxiliary dependencies `APP` for apposition and `KONJ` for conjunction are assigned by the parser. The order of dependency annotation is the following: `ROOT, APP, KONJ, SUBJ, OV, VPT, DOBJ, PRED, OD, OS, OG`.

In total, the dependency parser of GRIP comprises 1176 rules.

Morphological information, such as case information, represents a necessary basis for assignment of grammatical relations in German. Thus, if a clause contains a transitive verb, and a nominative and an accusative NP, it is safe to assume that a subject relation holds between the verb and the nominative NP and a direct object relation holds between the verb and the accusative NP. However, in real texts, the linkage between a case value and a function of a token is not always straightforward. Consider, for example, the sentence in (79):

(79) Julianne Köhler aber ist als sture, treue Musterdeutsche
Julianne Köhler however is as stubborn, loyal model German
eine Entdeckung.
a discovery.

'However, as a stubborn loyal model German, Julianne Köhler is a discovery.'

The sentence contains three nominative NPs, but only one of them is a subject of the verb: "*Julianne Köhler*". The noun phrase "*eine Entdeckung*" plays the role of a predicative object, whereas the noun phrase "*als sture, treue Musterdeutsche*" is a subject modifier.

For efficient and accurate dependency assignment, the following strategy is undertaken in the grammar: first, a dependency relation is established between any two nodes that can be connected by the relation. At this stage, the conditions only have reference to the following information:

- the categorial values of the nodes;
- the case value of the nodes (optionally);
- and what other nodes occur in the same clause.

For the example sentence (79), subject relations would be established between the verb and every nominative noun.

Next, more specific constraints on the context and the feature values of the tokens are stated in the grammar. According to these constraints, previously established dependencies can be eliminated or renamed. Thus, for example, since the first NP in sentence (79) agrees with the verb in number, all subject relations which involve other nominative NPs in the same sentence are renamed into predicative object relations. Next, the comparative NP "*als sture, treue Musterdeutsche*" is renamed from predicative object to subject modifier, since non-comparative predicative objects are preferred over comparative predicative objects.

Such a strategy of dividing the annotation process into two stages allows for the minimization of the set of dependency rules involved in parsing and for taking maximum advantage of the constraint-based nature of GRIP.

An example of a general rule which establishes relations of the first stage is provided in (80):

(80)
```
| VF{?*, NP{?*,noun#1[acc]; pron#1[personal,refl,dat:~];
pron#1[attr:~,acc]}}, ?*[lk:~,vf:~,sent:~,paren:~,spec:~,vc:~],
LK{?*,#2[verb,!aux:!]} | if (~DOBJ(#1,#2) & ~OV(#3,#2))
DOBJ(#1,#2).
```

In the rule, a context for the rule application is described between the pipe lines. It includes a sequence of nodes which starts with an initial topological field (`VF`) and ends with a left sentence bracket (`LK`). Between these two topological fields, any number of nodes can occur, unless the nodes bear features of topological fields `LK`, `VF` or `VC`, or features of either sentence-final punctuation, or parentheses, or a hyphen. The internal structure of the nodes `VF` and `LK` is explored and variables are assigned to the accusative head of a noun phrase and to a non-auxiliary finite verb. Thus, constraints are stated on the context, on the internal structure of the nodes and on features of lexical nodes. Moreover, constraints on previously established relations are imposed: nodes #1 (an accusative head of the NP in the initial field) and #2 (a finite verb occurring in the left sentence bracket) are required not to stand in a direct object relation with each other. Additionally, node #2 (a finite verb) is required not to have a verbal object. If all the constraints are satisfied, a direct object relation is established between nodes #1 and #2. An example sentence to which the rule would apply is provided in (81) below. A direct object relation would be established between the noun "*Kritik*" and the verb "*äußerten*".

(81)	Kritik	äußerten	hingegen	die	Bremer	Grünen.
	Criticism	expressed	however	the	Bremen	greens.

'However, the Bremen greens expressed a criticism.'

On the following stages, more specific constraints on the context and on other dependencies of the nodes are stated and relations violating the constraints are deleted or renamed. Thus, a rule in (82) insures that no node of a comparative construction participates in a direct object relation with a verb which has another direct object.

(82) | NP{conj[comparative],?*,?#1} | if (∧DOBJ(#1,#2) & DOBJ(#3,#2)) ~.

The rule (82) explores NPs which start with a comparative conjunction. It states that if any node inside such an NP participates in a direct object relation with some other node which has a second direct object, then the former direct object relation should be eliminated.

The strategy of establishing relations with general rules and the consequent elimination or renaming of relations with more specific constraints is used for the annotation of all dependencies in GRIP. Below, more details on the dependency annotation process are given for each dependency relation separately.

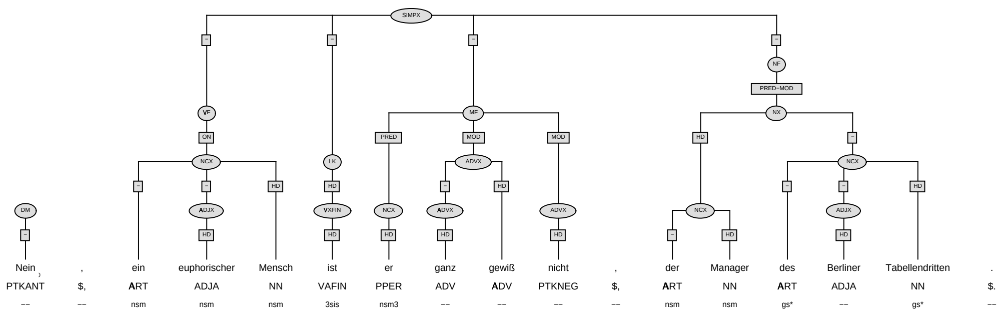

SIMPX
NF
PRED-MOD
VF
ON
NCX
LK
HD
MF
PRED
MOD
ADVX
NX
DM
ADJX
VXFIN
Nein , ein euphorischer Mensch ist er ganz gewiß nicht , der Manager des Berliner Tabellendritten .
PTKANT $, ART ADJA NN VAFIN PPER ADV ADV PTKNEG $, ART NN ART ADJA NN $.
nsm nsm nsm 3sis nsm3 nsm nsm gs* gs*

Figure 7.11: The treebank analysis for sentence (83)

Subject
As has been shown above, not all nominative NPs have the grammatical role of subject. Nominative NPs also serve as predicative objects ("*eine Entdeckung*" in sentence (79) above), as subject modifiers ("*als sture, treue Musterdeutsche*" in sentence (79) above), and as predicative object modifiers ("*der Manager des Berliner Tabellendritten*" in sentence (83)[3]).

(83) Nein, ein euphorischer Mensch ist er ganz gewiß nicht, der
No, an euphoric man is he quite surely not, the
Manager des Berliner Tabellendritten.
manager the$_{gen}$ Berlin$_{gen}$ third in the table
'No, he is definitely not an euphoric man, the manager of the Berlin team that placed third in the competition.'

Moreover, often apposition terms are expressed by a nominative form regardless of the function of the apposition NP, such as in sentence (84):

(84) Im Rahmen des von der Bürgerschaft beschlossenen
in the frame the$_{gen}$ by the township determined$_{gen}$
"Aktionsplans Alkohol" fand im Zentralkrankenhaus Ost
"action plan$_{gen}$ alcohol$_{nom}$" took in the central hospital east
eine Fachtagung zum Thema "Frauen und Alkohol" statt.
a symposium on the topic "women$_{nom}$ and alcohol$_{nom}$" place.
'Within the scope of the determined by the township "Alcohol action plan", a symposium on the subject "Women and alcohol" took place in the East central hospital.'

[3]The treebank analysis of (83) is presented in Figure 7.11.

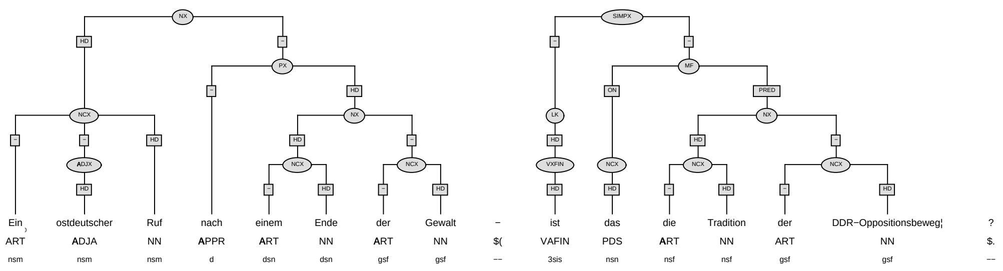

NX
HD
NCX
ADJX
PX
SIMPX
MF
ON
PRED
LK
VXFIN
Ein ostdeutscher Ruf nach einem Ende der Gewalt – ist das die Tradition der DDR−Oppositionsbeweg ?
ART ADJA NN APPR ART NN ART NN $(VAFIN PDS ART NN ART NN $.
nsm nsm nsm d dsn dsn gsf gsf -- 3sis nsn nsf nsf gsf gsf --

Figure 7.12: The treebank analysis for sentence (85)

Annotation of subject relations is further complicated by the presence of discourse markers and constructions that are analyzed as unattached subtrees in the treebank, such as in sentence (85) and its corresponding treebank analysis in Figure 7.12:

(85) Ein ostdeutscher Ruf nach einem Ende der Gewalt – ist
an Eastern German call for an end the$_{gen}$ violence$_{gen}$ – is
das die Tradition der DDR-Oppositionsbewegung?
it the tradition the$_{gen}$ GDR opposition movement$_{gen}$?
'An Eastern German call for an end to the violence – is that the tradition of the GDR opposition movement?'

The following constraints have been formulated on subject relations regardless of the presence of competing subject relations:

- if a modifier of the subject relation occurs in a comparative construction, the relation is renamed into a predicative object relation;
- if a noun phrase which contains the modifier of a subject relation expressed by a proper noun occurs immediately after a comparative NP, the relation is deleted;
- if a noun phrase which contains the modifier of a subject relation expressed by a proper noun is preceded by a coordinating conjunction and a comparative NP, the relation is deleted.

Further constraints apply if a verb participates in more than one subject relation as a head and if the modifiers of the corresponding subject relations are not connected by a coordination or an apposition relation. Among such constraints are:

- if one of the modifiers is an interrogative pronoun or a quantitative pronoun, the corresponding relation is renamed into a predicative object relation;
- in other cases, subject relation preference is given to personal pronouns and indefinite pronouns; other subject relations with the same head are renamed into predicative object relations;
- in cases of remaining ambiguity, subject relation preference is given to a relation whose modifier occurs in the leftmost position and agrees with the head verb in number; other relations are renamed into predicative object relations.

Subject relations can additionally involve as modifiers elements which are not marked with morphological case. Such elements include:

- foreign material: "Ein paar Running Gags werden zwar arg strapaziert." ("A couple of Running Gags get seriously over-used.");
- cardinals: "20 000 ist genug." ("20 000 is enough.");
- invariable pronouns: "Mehr werden es kaum." ("There hardly will be more.");
- subordinate clauses: "Daß die Ordnungsmuster ein wenig durcheinandergeraten sind, ist heute Standard." ("It is standard today that the order samples get a bit mixed up.")
- infinitival constructions: "Es wäre wahrscheinlich leichter, den debis-Mitarbeitern zu kündigen." ("To fire the debis employees would be probably easier.");
- prepositional phrases: "Über 20 000 ist besser." ("Above 20 000 is better.").

Annotation of relations which include invariable elements as modifiers is pursued if no morphologically marked subject has been found for a verb. The order in which the elements are presented above reflects preferences for choosing an invariable element as a subject if a clause contains more than one element of this kind: thus, invariable pronouns are preferred over subordinate clauses, infinitival constructions and prepositional phrases, whereas cardinals and foreign material tokens are preferred over invariable pronouns.

Direct object

Accusative noun phrases are as frequent, if not more frequent, than nominative noun phrases in German texts. Apart from playing the role of direct objects, accusative NPs serve as predicative objects ("*als "Boulevardkomödien""* in sentence (86) and "*vier Punkte*" in sentence (87)) and different kinds of modifiers (`OS-MOD` "*es*" in sentence (88), `V-MOD` "*nächste Woche*" in sentence (89)).

(86) Autor Oliver Bukowski, 1961 in Cottbus geboren, bezeichnet seine
author Oliver Bukowski, 1961 in Cottbus born, describes his
Stücke selbst als "Boulevardkomödien".
pieces himself as "boulevard comedies"
'Author Oliver Bukowski, born in Cottbus in 1961, describes his pieces himself as "boulevard comedies".'

(87) Der VfB war nur noch vier Punkte von einem Abstiegsplatz
The VfB was only yet four points from a relegation place
entfernt.
away
'The VfB was only four more points away from the relegation place.'

(88) Ich finde es gut, daß das Aktionsprogramm nun in einer breiten
I find it good, that the action program now in a wide
Öffentlichkeit diskutiert wird.
publicity discussed is.
'I find it good, that the action program is being widely discussed now.'

(89) Die SPD wird nächste Woche über das Thema beraten.
The SPD will next week about this subject discuss.
'The SPD will discuss this subject next week.'

An established direct object relation is renamed into a predicative object relation if a modifier of a direct object relation occurs in a comparative construction.

Examples of contexts in which an established direct object relation is eliminated are:

- a modifier of a direct object relation is a part of an NP which occurs in an initial topological field after a prepositional phrase (which means that the NP modifies the PP, since only one element can occur in the initial field);

- a modifier of a direct object relation has characteristics of time (such as the words "*Jahr*" ("year") or "*Montag*" ("Monday")) or frequency (such as "*Mal*" ("time"));
- a modifier of a direct object relation occurs in a passive clause.

Invariable tokens such as foreign words, cardinals, invariable pronouns and prepositional phrases can also play the role of direct objects. However, in the case of direct objects, identification of all contexts in which a relation with an invariable token should be established is more difficult than in the case of subjects. To avoid over-generation, grammar rules for establishing direct object relations with an invariable token are restricted to a small set and apply if an invariable token occurs in the final position of an accusative noun phrase (i.e., if it is a head of a noun phrase and if it has a preceding accusative determiner or modifier).

Dative object

Similar to the case of subject relation and direct object relation, comparative dative object relations are not allowed and are deleted by grammar constraints. Also, if a modifier of a dative object relation has characteristics of time or frequency, such as in the case of the token "*Ende April*" ("at the end of April") in sentence (90), the relation is renamed into a verbal modifier relation.

(90)

Der	19.	Zivilsenat	des	Düsseldorfer
The	19th	civil court of appeal	the$_{gen}$	Düsseldorf$_{gen}$

Oberlandesgerichts	befreite	Ende	April	den
Higher Regional Court$_{gen}$	released	end	April	the

Mannesmann-Konzern	von	dieser	Regelung.
Mannesmann combine	from	this	regulation

'The 19th civil court of appeals of the Higher Regional Court of Düsseldorf released the Mannesmann combine from this regulation at the end of April.'

The main two difficulties in the annotation of dative object relations are illustrated by sentences (91) and (92):

(91)

Endlich	nun	kam	der	Frankfurter	"Guru"	Berthold	Kilian	nach
Finally	now	came	the	Frankfurt	"Guru"	Berthold	Kilian	to

Bremen,	um	über	das	aktuell	diskutierte	Thema
Bremen,	in order	about	the	currently	discussed	subject

Co-Abhängigkeit zu sprechen.
co-dependency to talk

'The Frankfurt "Guru" Berthold Kilian finally came to Bremen to talk about the currently discussed subject Co-dependency.'

(92) "Wir müssen uns selbst helfen", meinte Magath.
"we must us ourselves help", meant Magath

'"We must help ourselves", said Magath.'

In the first case, the dative noun "*Co-Abhängigkeit*" is an apposition term and belongs to the noun phrase "*das aktuell diskutierte Thema Co-Abhängigkeit*". Such cases of a dative NP being an apposition term or a conjunct of a noun phrase included in a prepositional phrase are rather frequent in the newspaper texts used in the experiments. Therefore, the grammar should be able to identify such cases and avoid incorrect assignment of dative object relations to such dative NPs.

However, there are also cases in which a dative object relation should be assigned to a noun phrase in a similar context. Consider, for example, sentence (93) in which the noun phrase "*Besserem*" is a dative object of the verb "*weichen*":

(93) Sondern eher, weil der spießige "Bienenkorb" von Susi
But rather, because the narrow-minded "beehive" of Susi
Besserem weichen mußte.
the better$_{dat}$ give way had to.

'But rather because the narrow-minded "beehive" of Susi had to give way to something better.'

Structures like the one in sentence (93) are much less frequent than the structures similar to the one in sentence (92). Therefore, the following constraints apply in the GRIP grammar:

- if a modifier of a dative object relation stands in a coordination or an apposition relation with a token in a prepositional phrase, the dative object relation is deleted;

- if a modifier of a dative object relation is a part of a non-personal NP which occurs immediately after a prepositional phrase, the dative object relation is deleted.

Although the second heuristic constraint may lead to the elimination of correct relations, the overall performance of the system with such a heuristic is considerably higher than without it.

The second difficulty in the annotation of dative object relations concerns reflexive pronouns, such as "*sich*". German reflexive pronouns are invariable and are not assigned case information in the treebank analyses. Therefore, a distinction between the accusative and dative object uses of such pronouns is sometimes difficult to draw. The following heuristic constraints are used in GRIP for this problem:

- initially all relations involving a reflexive pronoun as a modifier are given a direct object name;
- if a head of the relation is an auxiliary verb, the relation is renamed into a dative object relation;
- if there is another direct object relation with the same head and if the verb is not a ditransitive verb, the reflexive direct object relation is renamed into a dative object relation.

To resolve residual cases, a list of verbs which take only a dative object and not a direct object is required.

Predicative object

During the annotation of subject and direct object relations, some of subject and direct object relations are renamed into predicative object relations. This renaming concerns in the first place relations which involve comparative constructions. However, not all comparative constructions play the role of predicative objects: consider a comparative construction "*als ehrenamtliche Vorsitzende*" in sentence (94) which modifies pronoun "*sie*" and is not a predicative object:

(94) Ute Wedemeier hält es für "selbstverständlich", daß sie als
Ute Wedemeier takes it for "granted", that she as
ehrenamtliche Vorsitzende ein dienstliches Handy hat.
volunteering chairperson an office mobile phone has.
'Ute Wedemeier takes it for "granted", that she as a volunteering chairperson should have an office mobile phone.'

The following heuristics of the GRIP grammar eliminate erroneous predicative object relations which involve comparative constructions:

- eliminate a relation if the NP which contains the modifier of the relation immediately follows the first NP in a field and if this first NP is marked as subject (this heuristic will eliminate an incorrect predicative object relation in sentence (94));
- eliminate a relation if the modifier of the relation is a nominative personal pronoun;
- eliminate a relation if the NP which contains the modifier of the relation immediately follows a predicative adjective or one of the following pronouns: "*mehr*" ("more"), "*weniger*" ("less"), "*anders*" ("different").

Apart from heads of comparative construction, predicative object relations can involve many other elements. Examples include:

- nominative nouns and pronouns: "*Entdeckung*" in sentence (79);
- accusative nouns and pronouns: "*Locken*" in sentence (95);
- dative nouns: "*Meinung*" in sentence (96);
- pronominal adverbs: "*dagegen*" in sentence (97);
- standard adverbs: "*so*" in sentence (98);
- prepositional phrases: "*in Ordnung*" in sentence (99);
- predicative adjectives: "*erreichbar*" in sentence (100);
- attributive adjectives: "*nächste*" in sentence (101);
- infinitival constructions: "*Ein Angebot formulieren*" in sentence (102);
- subordinate clauses: "*wie es ist*" in sentence (103).

(95) Und warum nennt er sie "Locken"?
And why calls he her "Curls"?
'And why does he call her "Curls"?'

(96) Ich bin deshalb immer noch der Meinung, dafür einen
I am therefore still the$_{dat}$ opinion$_{dat}$, instead a
speziellen Fonds zu haben.
special fund to have.
'Therefore, I am still of the opinion that it would be better to have a special fund instead.'

(97) Sein Kollege ist "strikt dagegen".
'His colleague is "strictly against it"'.

(98) Es ist so wie es ist.
It is so as it is.
'It is the way it is.'

(99) Das ist nicht in Ordnung.
It is not in order.
'It is unacceptable.'

(100) Sie ist nicht erreichbar.
She is not reachable.

(101) Wer wird die oder der nächste sein?
Who will the$_{fem}$ or the$_{masc}$ next be?
'Who will be the next one?'

(102) "Ein Angebot formulieren" nennt Hertwig dieses Vorgehen.
"an offer formulate" calls Hertwig this procedure.
'Hertwig calls this procedure "formulating an offer".'

(103) Methadonprogramm bleibt, wie es ist.
Methadone program remains as it is.
'The methadone program remains as it is.'

These examples show that tokens of almost any part of speech can theoretically play the role of predicative object. To avoid massive over-generation in the grammar, heuristic constraints which restrict the context are formulated in the grammar. Among them:

- predicative object relations which involve adjectives, prepositional phrases, adverbs, infinitival constructions and subordinate clauses are allowed only with auxiliary verbs and copula verbs, such as "*bleiben*" ("remain");
- predicative object relations which involve accusative tokens of non-comparative noun phrases are allowed only with "naming" verbs, such as "*nennen*" ("call", "name");
- if the modifier of a predicative object relation occurs after pronouns "*nichts*" ("nothing") or "*etwas*" ("something") then the relation is deleted, since in such cases the token tends to modify the pronoun:

(104) Ihre Kandidatur sei nichts Ungewöhnliches.
Her candidature be nothing unusual.
'Her candidature is claimed to be nothing unusual'.

- in the case of two predicative object relations with the same head, the one whose modifier occurs first in the sentence is deleted if:
  - it is an adjective and the second modifier is a pronominal adverb, an adjective, a head of a non-comparative noun phrase, or the last element in a field;
  - it is an adverb or an element of a prepositional phrase and the second modifier is a head of a non-comparative noun phrase;
- delete a predicative object relation if its modifier is followed by a modifier of a direct object relation with the same head, by a modifier of a subject relation with the same head, or by an adverb.

Genitive object

A genitive object relation is initially established between verbs which can take a genitive object, such as "*achten*" ("respect"), "*sich erinnern*" ("remember"), "*anklagen*" ("accuse"), "*bedürfen*" ("require"), "*versichern*" ("ensure") etc., and genitive nouns.

At the second stage, grammar constraints consider verbs which are assigned two genitive object relations and eliminate one of them if:

- an NP containing a genitive noun immediately follows another NP, or
- an NP containing a genitive noun immediately follows a genitive prepositional phrase, or
- an NP containing a genitive noun is preceded by a coordinating conjunction and a genitive prepositional phrase, or
- an NP containing a genitive noun occurs in a non-initial position in the initial field.

In all these situations the genitive NP tends to be a genitive modifier rather than a genitive object.

Further constraints are stated on reflexive and separable verbs which take genitive objects. Such verbs can participate in a genitive object relation only if a reflexive pronoun or a separable prefix, respectively, occurs in the same clause. The reason for stating such constraints can be illustrated with the example of the German verb "*sich annehmen*" ("take care", "look after"), which is both separable and reflexive. Consider sentence (105):

(105)

Nach	Angaben	Naumanns	nehmen	sich	bereits
According to	information	Naumann$_{gen}$	take	themselves	already

heute	privat	organisierte	Einrichtungen	verfolgter	Künstler	an.
today	privately	organized	institutions	persecuted	artists	on

'According to Naumann's information, already today privately organized institutions are taking care of persecuted artists.'

Here, a genitive object relation holds between the verb "*nehmen*" and the noun "*Künstler*". Since neither a non-reflexive separable verb "*annehmen*" nor a non-reflexive non-separable verb "*nehmen*" take a genitive object, constraints are needed that ensure that a genitive object relation cannot be assigned to the verb "*nehmen*" unless both a prefix "*an*" and a reflexive pronoun occur in the clause.

Verbal object

A verbal object relation holds between elements in a verbal chain. Initially, a set of rules determines the context in which two verbs or a verb and a particle "*zu*" can be connected by such relation: either the elements of the relation belong to the same verbal complex, or one of the elements is a finite verb in the left sentence bracket while the second element occurs in the verbal complex of the same clause.

However, not only a verbal object relation can hold between two such elements. Compare, for example, sentences (106) and (107):

(106)

Meine	Aussage	im	Interview	ist	nur	im	Zusammenhang	mit
my	statement	in the	interview	is	only	in	connection	with

dem	Inhalt	der	gestellten	Fragen	zu	verstehen.
the	content	the$_{gen}$	asked$_{gen}$	questions$_{gen}$	to	understand.

'My statement in the interview should be understood only in connection with the content of the questions asked.'

(107)

Jamal	beginnt	zu	verstehen.
Jamal	begins	to	understand.

In sentence (106), a verbal object relation connects the verbs "*ist*" and "*verstehen*". In sentence (107), the verbs "*beginnt*" and "*verstehen*" are linked by a sentential object relation. The general strategy for distinguishing these two types of relations relies on the notion of *coherency* and on the type of the finite verb in terms of coherency. Three types of verbs are distinguished according to Bech (1955-57):

1. verbs constructing coherently and incoherently, e.g. "*versprechen*" ("promise"), "*versuchen*" ("try");

2. verbs constructing only coherently, e.g. "*wollen*" ("want"), "*möchten*" ("like");

3. verbs constructing only incoherently, e.g. "*überreden*" ("persuade"), "*überzeugen*" ("convince").

Verbs constructing only coherently always build a verbal object relation with other verbs, whereas verbs constructing incoherently take a sentential object as an argument. Based on this regularity and a list of incoherently constructing verbs in the grammar lexicon, constraints are formulated for differentiating `OS` and `OV` relations.

Sentential object

Sentential object (`OS`) relations are initially established between a finite verb of a main clause and a finite verb of a subordinate clause or a main verb in a verbal complex. Subordinate clauses are restricted to clauses of non-adverbial meaning, i.e. purpose, time, condition, and reason subordinate clauses are excluded from consideration.

Sentential object relations can be mixed with verbal object relations (see above in the description of the rules for verbal object relations), subject relations between a subordinate clause and a verb (see sentence (108)) and with different kinds of modification relations (e.g. `OPP-MOD` in sentence (109)).

(108) Deshalb bleibt es sein Geheimrezept, wie er es schafft.
Therefore remains it his secret recipe, how he it manages.
'Therefore how he manages it remains his secret recipe.'

(109) Denn auch die gehen davon aus, daß sie ohne das
because also those go therefrom, that they without the
BLG-Monopol preiswerter arbeiten könnten.
BLG monopoly less expensive work could.
'Because they also assume that without the BLG monopoly they could work more cheaply.'

Since verbal objects and subjects are annotated prior to annotation of sentential objects, a constraint is stated in the grammar that prohibits establishing a sentential object relation between two tokens if they are connected by a subject or a verbal object relation.

To distinguish sentential objects from modifiers, the following heuristics are used:

- if a subordinate clause is preceded by an adverbial pronouns, the `OS` relation is deleted; the constraint does not apply if the adverbial pronoun has a form "*trotzdem*" or "*daraufhin*";
- if the head of an `OS` relation is an auxiliary verb, the relation is deleted.

Separable verbal prefix

Due to the fact that separable verbal prefixes have a specific categorial tag, annotation of this dependency relation is rather simple. The relation is established between a rightmost finite non-auxiliary verb in a left bracket field and a separable verbal prefix in a verbal complex. The main difficulty in this case is to assign a relation between a correct pair of tokens, since the verbal complex can occur both after and in front of the left bracket.[4] Consider, for example, sentence (110) which presents an example of a verbal complex preceding the finite verb:

(110) Dem	Schmutz	gegenüber	steht	in	polnisch-katholischer	Manier
the$_{dat}$	scum$_{dat}$	opposite	stands	in	Polish Catholic	manner
das	Engelsgesicht	von	Antonina.			
the	angelic face	of	Antonina			

'The angelic face of Antonina confronts the scum in a Polish Catholic manner.'

The analysis of sentence (110) is presented in Figure 7.13.

7.4.4 Extraction of dependency information from Tüba-D/Z

For the evaluation of the parser, dependency relations have been extracted from the treebank and presented in a form compatible with the parser output, i.e. as a set of triples $< M, H, L >$, where H is a head element, M is a modifier and L is a dependency label. Since in the treebank dependency

[4]Although, of course, the latter case is rather rare.

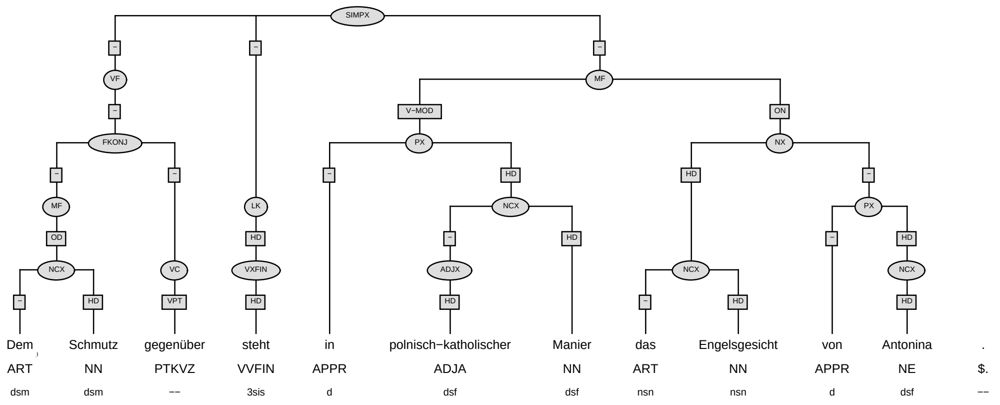

SIMPX
VF
FKONJ
MF
OD
NCX
VC
VPT
LK
HD
VXFIN
MF
V−MOD
PX
NCX
ADJX
ON
NX
PX
NCX
Dem Schmutz gegenüber steht in polnisch−katholischer Manier das Engelsgesicht von Antonina .
ART NN PTKVZ VVFIN APPR ADJA NN ART NN APPR NE $.
dsm dsm −− 3sis d dsf dsf nsn nsn d dsf −−

Figure 7.13: The treebank analysis for sentence (110)

relations are encoded indirectly, i.e. as edge labels in a constituency tree, the procedure for dependency extraction is described below.

Extraction of dependency relations is equivalent to identification of a head (`HEAD`) and a dependency label (`DEP-LABEL`) for each lexical token. The procedure differs for lexical nodes which are marked as head elements in a phrase (i.e. nodes, whose ascending edge is marked with "`HD`", like that of a node "*Arbeit*" in Figure 7.14) and for lexical nodes which play the role of a modifier (their ascending edge is marked either with a label of a non-head daughter "`-`" or with a name of a grammatical function, such as "`ON`").

For a non-head lexical node, the head element `HEAD` is identified as a lexical head of its head sister. Thus, in Figure 7.14, `HEAD` of node "*Die*" is a lexical node "*Arbeit*". The label of the dependency relation that connects a non-head lexical token and its head (`DEP-LABEL`) is encoded in the function of the node, i.e. in the label of the ascending edge of the node (label "-" for node "*Die*" in Figure 7.14).

To establish a dependency of a lexical node which plays the role of the head in a phrase, the first step is to find the lowest non-head ancestor of the node. For node "*zu*" in Figure 7.14 such an ancestor is the `SIMPX` node with the function `OS` (node number 509). The function of the ancestor contains a dependency label for the lexical node. `HEAD` of the lexical node corresponds to a lexical head of a head sister of the lowest non-head ancestor of the node.

A special case is represented by constructions which do not contain a head element, such as coordination and apposition, as well as sentential nodes. In the procedure described above, all conjuncts and apposition terms

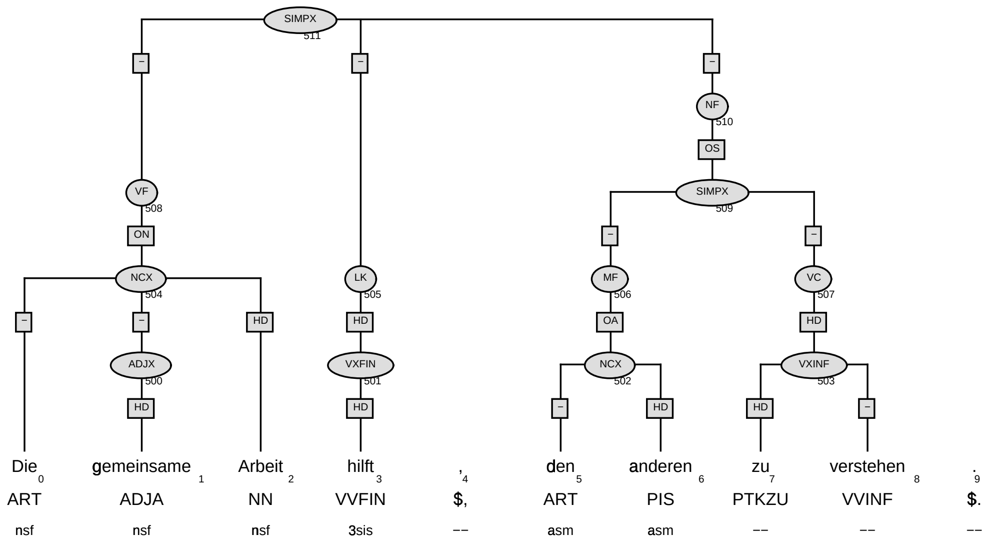


Figure 7.14: An example TüBa-D/Z tree

are treated as head elements. In a sentential node, the head element is identified as a lexical head of `LK` node, which stands for the left bracket of the sentence and contains a finite verb. If the sentence does not contain a left bracket, the sentential head is identified as a lexical head of a verbal complex `VC`.

A lexical node which does not have a governing element is marked as the root of the sentence (`ROOT`).

Table 7.6 lists dependencies extracted from the tree presented in Figure 7.14.

7.4.5 Evaluation

The performance of the dependency module run on the data with correct morpho-syntactic tags is presented in Table 8.1. Prior shallow constituency analysis of the data is provided to the dependency module by the GRIP chunker. The metrics of labeled and unlabeled precision, recall and f-measure described above have been used for the evaluation.

$$(Labeled)\ Precision = \frac{\#matched\ dependencies}{\#dependencies\ in\ test\ data}$$

Dependency	Modifier	Head
-	Die	Arbeit
-	gemeinsame	Arbeit
ON	Arbeit	hilft
ROOT	hilft	
-	den	anderen
OA	anderen	verstehen
OS	zu	hilft
-	verstehen	zu

Table 7.6: Dependencies extracted from the tree in Figure 7.14

$$(Labeled)\ Recall = \frac{\#matched\ dependencies}{\#dependencies\ in\ gold\ data}$$

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

Dep.	**Labeled**			**Unlabeled**		
label	**Recall**	**Prec.**	**F-meas.**	**Recall**	**Prec.**	**F-meas.**
total	94.91	94.55	94.73	95.95	95.57	95.76
ROOT	98.12	97.71	97.91	98.16	97.76	97.96
SUBJ	96.28	95.50	95.89	96.36	95.56	95.96
DOBJ	93.08	91.38	92.22	93.08	91.38	92.22
OD	83.69	83.15	83.42	83.69	83.15	83.42
OG	100	100	100	100	100	100
OS	70.74	71.27	71.00	71.27	71.63	71.45
OV	97.29	97.11	97.20	97.29	97.11	97.20
PRED	69.58	70.83	70.20	69.58	70.83	70.20
VPT	98.46	98.46	98.46	98.46	98.46	98.46

Table 7.7: Evaluation of the GRIP dependency parser

7.4.6 Error analysis

The current subsection describes the most frequent errors made by the dependency parser in the assignment of dependencies.

In the annotation of a **subject** relation, the most frequent error is failure to identify the relation due to a complicated sentence structure. Often such complicated structures involve complex named entities, such as the citation of a statement "*Diskussionen um Menschenrechtsverletzungen spielen weder bei den Spendern noch bei den Hilfsangeboten eine Rolle*" in sentence (111), the occurrence of sentence-final punctuation inside a clause, such as the occurrence of an exclamation mark in sentence (112), or complex embedded constructions.

(111) Meine Aussage im Interview "Diskussionen um
my statement in the interview "discussions about
Menschenrechtsverletzungen spielen weder bei den Spendern noch
violations of human rights play neither by the donators nor
bei den Hilfsangeboten eine Rolle" ist nur im Zusammenhang mit
by the help offers a role" is only in connection with
dem Inhalt der gestellten Fragen zu verstehen.
the content the$_{gen}$ asked$_{gen}$ questions$_{gen}$ to understand
'My statement in the interview "Discussions about violations of human rights play no role either for the donators or for the help offers" should be understood only in connection with the content of the questions asked.'

(112) Man mag kaum glauben, daß die Hauptdarstellerin Naomi (!)
One could hardly believe, that the leading actress Naomi (!)
Nishida in Japan ein gefeiertes Fotomodell ist.
Nishida in Japan a famous photographic model is
'One could hardly believe, that the leading actress Naomi (!) Nishida is a famous model in Japan.'

Another source of errors is failure to annotate subjects that do not have any morphological features, such as foreign material tokens, cardinals or prepositional phrases. Errors are also caused by incorrect chunking analysis provided for the dependency parser and by confusion of the relations which usually involve nominative lexemes: subjects, predicative objects and the modifiers of subjects.

The last prominent group of subject errors concerns annotation of subjects which represent apposition terms. According to the annotation style of the parser, if an apposition group participates in a dependency relation, then all terms of apposition are assigned the dependency relation in question. However, in some cases correct identification of the apposition construction is complicated by the complex structure of the sentence. In such cases, only one of the apposition terms is assigned a dependency relation, since multiple subjects are ungrammatical in German.

Errors in the annotation of a **direct object** relation partly coincide with the errors made in subject relations. Complicated sentence structures and apposition constructions, chunking errors and absence of morphological features lead to problems in assignment of direct object relations. An error which is specific for the annotation of this dependency is a confusion of the grammatical function of the reflexive pronoun "*sich*" which does not inflect and, therefore, has the same form when used as a direct and indirect object. To annotate the pronoun correctly, subcategorization information is needed.

The same problem leads to decreased precision in the annotation of **dative objects**. Together with the problem of erroneous chunking it represents the main source of errors for the annotation of a dative object dependency.

Genitive objects are rather rare in the Tüba-D/Z data. The parser has captured all the genitive object relations in the test data and scored 100% accuracy.

A rather frequent error in the annotation of **sentential object** occurs when a parser is unable to recognize the coordination of clauses without a conjunction, such as in sentence (113):

(113) Tegeler bestätigt den Vorgang der Provisionszahlungen, meint
Tegeler confirms the procedure of the provision payments, thinks
allerdings, es müsse ein "Buchungsfehler" gewesen sein.
however, it must a "booking error" been be
'Tegeler confirms the procedure of provision payments, but thinks that there must have been a booking error.'

Another common error in the annotation of sentential objects concerns the treatment of discourse markers. They represent unattached nodes in the treebank but are part of the structure provided by GRIP. Consider the Tüba-D/Z analysis of sentence (114) provided in Figure 114. In the GRIP analysis, the discourse marker "*raunte ein Zuschauer*" is annotated as a part of the initial field of the sentence "*müssen wir jetzt aufstehen?*," which leads to the erroneous establishment of a dependency relation between the verb "*raunte*" and the verb "*müssen*".

(114) "Das ist ja wie vor Gericht", raunte ein Zuschauer, "müssen
"This is - as in court", murmured a spectator, "must
wir jetzt aufstehen?"
me now stand up?"

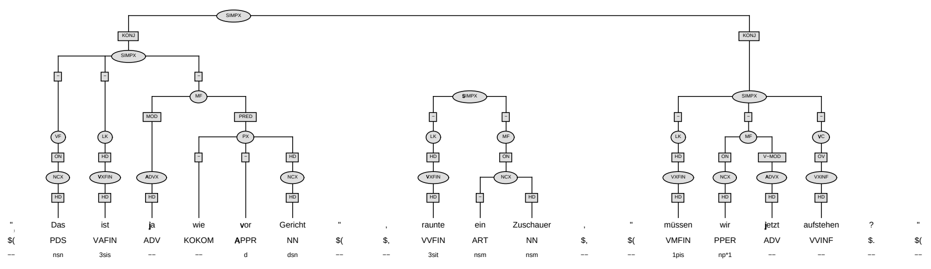

SIMPX
KONJ
LK
VF
MF
VC
PRED
MOD
PX
NCX
VXFIN
ADVX
VXINF
ON
OV
V-MOD
HD
" Das ist ja wie vor Gericht " , raunte ein Zuschauer , " müssen wir jetzt aufstehen ? "
$(PDS VAFIN ADV KOKOM APPR NN $($, VVFIN ART NN $, $(VMFIN PPER ADV VVINF $. $(
-- nsn 3sis -- -- d dsn -- -- 3sit nsm nsm -- -- 1pis np*1 -- -- -- --

Figure 7.15: An example TüBa-D/Z tree with a discourse marker

'"Is this as in court", murmured a spectator, "do we have to stand up now?"'

The same problem of the erroneous treatment of discourse markers causes one of the most frequent errors in the annotation of the **root elements**. According to the Tüba-D/Z treebank, sentence (114) has three roots: verbs "*ist*", "*raunte*" and "*müssen*". GRIP has assigned a single root "*müssen*" to this sentence. Other errors in the annotation of this dependency are also due to incorrect constituency analysis provided by the chunker.

The annotation of **verbal objects** is also highly influenced by the constituency analysis of data. A verbal object dependency is usually established between the finite verb of a sentence and a verb in a verbal complex. Errors in the constituency analysis complicate the annotation of the dependency. Another source of errors in this category concerns the confusion of verbal and sentential objects.

The main source of errors which leads to a decreased precision in annotation of **predicative objects** concerns comparative constructions. Comparative constructions often have the function of predicative objects in a sentence, such as in example (115). However, they also can serve as modifiers of other phrases, such as in example (116). Since the predicative object function is more frequent, the parser annotates comparative construction as such. To avoid the errors, semantic information would have to be taken into account.

(115)

Doch	dann	wurde	die Erstplazierte
but	then	became	the one who took the first place

überraschenderweise	als	mutmaßliche	Hetera	enttarnt.
surprisingly	as	probable	Hetera	exposed

'But then the winner was surprisingly exposed as a probable Hetera.'

(116) Ich hatte keine Lust, auf den Lesungen als Alphamännchen eine
I had no desire, on the readings as little alpha-man an
Ayse-Fibel darzubieten.
Ayse-primer to present
'I had no desire to present as a little alpha-man an Ayse-primer on the readings.'

Errors which leads to a decreased recall in the annotation of predicative objects include failures to establish a relation between adjectives and non-auxiliary verbs, such as in case of a phrase "*schlecht denken*" ("*to think bad*") and failures to establish a relation between prepositional phrases and verbs, such as in case of "*sich in guter Verfassung fühlen*" ("*to feel oneself in a good state*"). The introduction of corresponding rules into the grammar leads to a significant drop in precision, since relations between such elements are rather rare.

Errors in the annotation of **separable verb suffixes** are caused exclusively by the wrong annotation of subordinate clauses. In cases where a verbal complex is included in a wrong clause, the correct establishment of a separable verb suffixes relation is problematic.

7.5 Conclusion

The current chapter has introduced the GRIP dependency parser, a robust rule-based deterministic parser for annotation of German. Unlike standard dependency parsers which incorporate a rich lexicon with detailed subcategorization and semantic information into the grammar, the GRIP parser relies mainly on morpho-syntactic information for the annotation of dependency structures, limiting the use of subcategorization information to a few features for high frequency verbs.

The parser has been evaluated against the Tüba-D/Z treebank data. It has demonstrated state-of-the-art performance of 95.74% labeled f-measure. These high results indicate that correct annotation of morpho-syntactic information can serve as a sufficient basis for a successful dependency analysis.

Chapter 8

The GRIP system

The current chapter provides an overview of the complete GRIP system.

8.1 GRIP architecture

The complete GRIP system consists of a morphological analyzer, a morpho-syntactic disambiguator, a chunker and a dependency parser for German. All modules except for the morphological analyzer have been developed in the current dissertation project and are described in the relevant chapters above. The system can be used for the tasks of tagging, chunking and/or dependency parsing and allows for external input of different structure on any level of annotation.

The system is robust and deterministic: for every input sentence, at most one analysis is output. If no complete analysis can be provided due to unknown phenomena occurring in the input, the system supplies a partial analysis.

Figure 8.1 schematizes a general architecture of the system.

In GRIP, all possible morphological analyses for input tokens are provided by the Xerox morphological analyzer for German developed at the Xerox Research Centre Europe. The analyzer is described in more detail in Section 6.1.1. The analyses are then passed to the morpho-syntactic tagging module.

The disambiguation procedure starts with the resolution of part-of-speech ambiguity by a rule-based module (see section 6.1.3) and is followed by a rule-based module for the disambiguation of morphological values (see section 6.1.4). The residual ambiguity left by the rule-based modules is resolved by a statistical module based on PCFGs (see section 6.2.2).

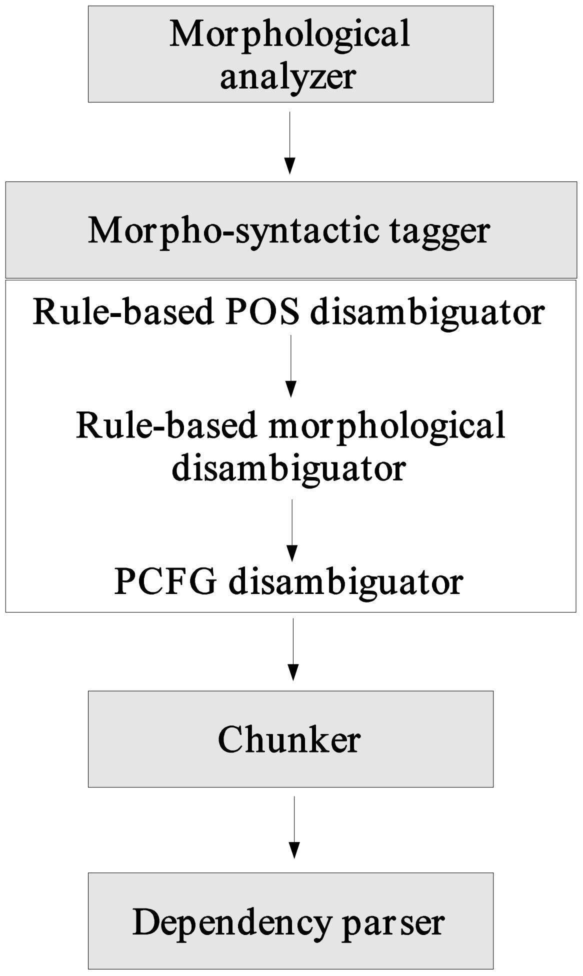


Figure 8.1: Architecture of the GRIP system

The GRIP chunker provides a shallow syntactic analysis for the input string based on the part-of-speech categories of tokens. Apart from the identification of the phrasal categories `AP` (adjectival phrase), `NP` (noun phrase) and `PP` (prepositional phrase), it also analyzes topological fields and subordinate clauses. The chunker is described in section 7.3.

The dependency module of GRIP expects a pre-chunked input with (possibly ambiguous) morpho-syntactic characteristics of tokens. The module provides annotation for verbal complements. More details on the module can be found in section 7.6.

8.2 GRIP evaluation

The complete system performance has been evaluated on 12 020 tokens from the treebank. The system has been provided with the tokenized input. Table 8.2 presents the results of the evaluation of the system. Statistics on labeled precision, recall and f-measure are provided for each module.

Module	Precision	Recall	F-measure
Tagger	89.33 %	91.29 %	90.30 %
Chunker	92.90 %	91.79 %	92.34 %
Parser	86.05 %	85.06 %	85.55 %

Table 8.1: Evaluation of the GRIP dependency parser

Error analyses for each module are provided in the corresponding sections above: section 6.4.3 for the tagger, section 7.3.4 for the chunker and section 7.4.6 for the dependency parser. Since errors are cumulative in case of the complete system, chunking and parsing performance is lower than when the chunker and the parser are run on gold input. New chunking and parsing errors arise from errors in tagging.

Table 8.2 provides evaluation of the speed of the system. Two metrics are used for the evaluation. The first metric estimates number of tokens processed by the system per second. This estimate is presented in column 2 (`tokens/sec`) of the table. The second metric evaluates the speed of the system in terms of sentences and represents the amount of time which is required by the system for the analysis of one sentence. The results for this metric are presented in column 3 (`sec/sent`) of the table.

The system performance was evaluated on a Sun SunBlade 100.

The table provides an evaluation of each separate module of the system as well as of the performance of the complete system. Since the GRIP morpho-syntactic tagger itself has a modular architecture with components of different nature, it is instructive to consider the speed of each component separately. The line "Rule-Based" stands for the performance of the GRIP rule-based disambiguator. The line "Mapping" represents results for a Perl-based mapping component which brings data output by the rule-based component in the format required by the following PCFG-based tagging module. Speed of the PCFG tagger is provided in line "PCFG".

Module	Speed	
	(tokens/sec)	(sec/sent)
Tagger	6.26	2.65
Rule-Based	212.93	0.08
Mapping	485.48	0.03
PCFG	6.85	2.4
Chunker	418.52	0.04
Parser	713.94	0.02
Total	6.11	2.71

Table 8.2: Speed of the GRIP dependency parser

As Table 8.2 shows, the PCFG module represents the slowest module of the system. However, it should be noted that the speed of the PCFG tagger varies greatly depending on the length of the sentence. Thus, tagging a sentence of an average test data length of 15 tokens with the PCFG module takes less than a second, whereas some sentences may take as long as 30-50 seconds. An example of such a sentence is provided in (117) below:

(117)

Laura Marina	Mit	seinen	skurrilen	Gestalten,	grandiosen	
Laura Marina	with	his	bizarre	shapes,	terrific	
Gesichtern	und	unbezahlbarem	Witz	erzählt	Regiseur	Kirk Jones
faces	and	invaluable	wit	tells	director	Kirk Jones

eine	Geschichte	aus	dem	Leben,	voller	Herz	und	natürlich	mit
a	story	from	the	life,	full of	heart	and	of course	with

einem	tiefen	Blick	in	menschliche	Abgründe.
a	deep	look	at	human	abysses.

Apart from the sentence length (34 tokens), three phenomena make the sentence complicated for the PCFG module to parse. First, the sentence includes a proper name "*Laura Marina*" which should belong to a separate sentence. Secondly, the sentence contains a misspelled word "*Regiseur*"[1]. Lastly, five words in the sentence ("*Laura*", "*Marina*", "*Regiseur*", "*Kirk*" and "*Jones*") represent unknown words and are therefore initially assigned the rather extensive set of all possible analyses.

As compared to the speed of the best tagger for German, TnT, which tags the sentence in less than a second, the PCFG module is much slower. However, this drawback is compensated by a clear advantage in qualitative results: whereas TnT reaches an accuracy of only 79.41%, the PCFG module provides an accuracy 94.12%, the only tagging mistakes being incorrect analyses for the misspelled token and for the adjective "*voller*" (`ADJA.d` instead of `ADJD`).

[1]Correct form: "*Regisseur*".

Chapter 9

Conclusion

The current thesis has been concerned with a problem of morpho-syntactic annotation of German.

Different approaches to tagging have been reviewed in the thesis. Two of the approaches have been selected for the development of a German tagger based on the characteristics of the approaches and their applicability to the specific task of morpho-syntactic annotation of German. The selected approaches are a rule-based approach and an n-gram-based approach.

A rule-based tagger has been developed and evaluated on German data. It has been shown that the rule-based tagger provides high accuracy of annotation and significantly reduces massive ambiguity of tokens. However, for 30% of tokens, the tagger fails to resolve the ambiguity completely.

Experiments with an n-gram tagger performed in the dissertation project have demonstrated insufficiency of the n-gram method for the task in question. The main problem of the method concerns a restricted context window taken into account in the annotation process, which turns insufficient for correct annotation of morphological features. A probabilistic method which takes a more global context into account, namely probabilistic phrase-structure grammars (PCFGs), has been chosen as a basis for an advanced tagging model. In order to take the best advantage of the strengths of the PCFG-based tagger, systematic transformations have been introduced to the structure of the training data, which has significantly improved the tagger performance.

Finally, a combined tagging model with a rule-based and a PCFG-based modules has been developed. The model brings together the powers of the two methods involved: based on the context, the rule-based module reduces lexical ambiguity and provides high accuracy of annotation. The subsequent

PCFG module resolves ambiguity completely based on statistics collected from training data.

The evaluation of the combined model has demonstrated the high performance of the model which beats the performance of the modules used independently and also exceeds the results of the best morpho-syntactic tagger for German described in the literature.

For a task-oriented evaluation of the combined tagging model, the model has been incorporated into a parsing system. A rule-based dependency parser which relies mainly on morpho-syntactic characteristics and on the linear order of input tokens has been developed. The parser has demonstrated high performance.

Experiments with the parser run on input with correctly annotated morpho-syntactic information have shown the state-of-the-art performance of the parser and confirmed the claim about the importance of morpho-syntactic information in parsing.

9.1 Contributions

The main contributions of the thesis are summarized below:

- A morpho-syntactic tagger which provides state-of-the-art performance even when trained on modest sets of data has been developed and described in the thesis. The tagger outperforms the best morpho-syntactic tagger for German described in the literature by 5.9%.

- A novel method of tagging with probabilistic phrase-structure grammars has been described together with techniques which lead to a significant improvement of the performance of a PCFG-based tagger.

- A rule-based dependency parser for German has been developed.

- Evaluation of the parser has demonstrated that state-of-the-art parsing performance can be achieved if a parser is provided with morpho-syntactic information for tokens. Such a parser does not require detailed information about subcategorization frames and semantic features of tokens and, therefore, avoids the bottleneck problem of knowledge acquisition.

- Experiments with the parser have confirmed the claim about importance of morpho-syntactic information in parsing.

9.2 Future Research

Two main areas of future research can be identified according to the two foci of the thesis: morpho-syntactic annotation and dependency parsing.

In the area of **morpho-syntactic annotation**, one of the possible research directions is to separate the annotation process into two steps: a part of speech annotation step and a consequent step of morphological annotation. This strategy seems intuitive, since the search space on each step is restricted to subsets of the possible analyses, which simplifies the task of the tagger. Moreover, annotation of POS categories hardly depends on morpho-syntactic information, which means that no clues would be lost for the tagger in case of a split process. Therefore, such strategy should lead to an improved tagging performance. The strategy has already been applied in the GRIP rule-based tagger and proved successful.

In the **dependency parsing** area, an interesting question continuing the research line of the thesis would be to investigate the issue of chunking as a pre-processing step to the parsing module. In the GRIP system described in the thesis, annotation of dependencies relies on prior chunking analysis provided for the input string. However, in classical dependency theory, no reference to constituencies is made and the analysis is based exclusively on the characteristics of tokens. It would be instructive to research whether a successful dependency parser which does not require any subcategorization or semantic information and relies only on morpho-syntactic characteristics of tokens and a linear order of tokens can be built without the incorporation of a preprocessing chunking module.

Another obvious way to proceed would be to extend the scope of the dependency parser to the annotation of other dependencies, such as verbal adjuncts and non-verbal arguments.

Appendix A

Stuttgart-Tübingen Tagset (STTS)

No.	Tag	Description
1	ADJA	attributive adjective
2	ADJD	predicative or adverbial adjective
3	ADV	adverb
4	APPR	preposition; circumposition, left part
5	APPRART	preposition with an article
6	APPO	postposition
7	APZR	circumposition, right part
8	ART	article
9	CARD	cardinal
10	FM	foreign material
11	ITJ	interjection
12	KOUI	subordinating conjunction with *zu* and infinitive
13	KOUS	subordinating conjunction with a clause
14	KON	coordinating conjunction
15	KOKOM	particle of comparison, no clause
16	NE	proper noun
17	NN	common noun
18	PDS	substituting demonstrative pronoun
19	PDAT	attributive demonstrative pronoun
20	PIS	substituting indefinite pronoun
21	PIAT	attributive indefinite pronoun without determiner
22	PIDAT	attributive indefinite pronoun with determiner
23	PPER	irreflexive personal pronoun

No.	**Tag**	**Description**
24	PPOSS	substituting possessive pronoun
25	PPOSAT	attributive possessive pronoun
26	PRELS	substituting relative pronoun
27	PRELAT	attributive relative pronoun
28	PRF	reflexive personal pronoun
29	PWS	substituting interrogative pronoun
30	PWAT	attributive interrogative pronoun
31	PWAV	adverbial interrogative or relative pronoun
32	PROP	pronominally used preposition
33	PTKZU	"zu" with infinitive
34	PTKNEG	negation particle
35	PTKVZ	separated verb particle
36	PTKANT	answer particle
37	PTKA	particle with adjective or adverb
38	TRUNC	truncated word, first part
39	VVFIN	finite main verb
40	VVIMP	imperative, main
41	VVINF	infinitive, main
42	VVIZU	infinitive with "zu", main
43	VVPP	past participle, main
44	VAFIN	finite auxiliary verb
45	VAIMP	auxiliary imperative
46	VAINF	auxiliary infinitive
47	VAPP	auxiliary past participle
48	VMFIN	finite modal verb
49	VMINF	modal infinitive
50	VMPP	modal past participle
51	XY	non-word containing special characters
52	$,	comma
53	$.	sentence-final punctuation
54	$(	sentence-internal punctuation

Appendix B

XRCE morphological analyzer

B.1 POS tags

No.	Tag	Description	Corresponding STTS tag
1	ADJA	(positive) attributive adjective	ADJA
2	ADJA2	comparative attributive adjective	ADJA
3	ADJA3	superlative attributive adjective	ADJA
4	ADJD	(positive) predicative or adverbial adjective	ADJD
5	ADJD2	comparative predicative or adverbial adjective	ADJD
6	ADJD3	superlative predicative or adverbial adjective	ADJD
7	ADV	non-adjectival adverb	ADV
8	ART	article	ART
9	CARD	cardinal	CARD
10	CIRCP	circumposition, right part	APZR
11	CM	comma	$,
12	COADV	adverbial conjunction	KON ADV
13	COALS	conjunction "als"	KOKOM KOUS
14	COINF	infinitival conjunction	KOUI

No.	**Tag**	**Description**	**Corresponding STTS tag**
15	COORD	coordinating conjunction	KON
			KOKOM
16	COP1	coordination 1st part	KON
17	COP2	coordination 2nd part	KON
18	COSUB	subordinating conjunction	KOUI
			KOUS
19	COWIE	conjunction "wie"	KOKOM
			KOUS
20	DATE	date	CARD
21	DEMADJ	demonstrative adjective	PDS
			PDAT
			ADJA
			PIDAT
			PIS
22	DEMDET	demonstrative determiner	PDAT
23	DEMINV	invariant demonstrative	PIDAT
24	DEMPRO	demonstrative pronoun	PDS
25	FM	foreign material	FM
26	INDADJ	indefinite adjective	PIS
			PIDAT
			PIAT
27	INDDET	indefinite determiner	PIAT
			PIDAT
28	INDINV	invariant indefinite	PIDAT
29	INDPRO	indefinite pronoun	PID
30	ITJ	interjection	ITJ
31	NOUN	noun	NN
			NE
32	ORD	ordinal	NN
			ADJA
33	PERSPRO	personal pronoun	PPER
34	POSDET	possessive determiner	PPOSAT
35	POSPRO	possessive pronoun	PPOSS
36	POSTP	postposition	APPO
37	PREP	preposition	APPR
38	PREPART	preposition with an article	APPRART
39	PROADV	pronominal adverb	PROP

No.	Tag	Description	Corresponding STTS tag
40	PTKANT	sentential particle	PTKANT
41	PTKCOM	comparative particle	PTKA
42	PTKINF	particle: infinitival “zu”	PTKZU
43	PTKNEG	particle: negation “nicht”	PTKNEG
44	PTKPOS	positive modifier	PTKA
45	PTKSUP	superlative modifier	PTKA
46	PUNCT	sentence internal punctuation	$(
47	REFLPRO	reflexive “sich”	PRF
48	RELPRO	relative pronoun	PRELS
			PRELAT
49	REZPRO	reciprocal “einander”	PRF
50	SENT	sentence final punctuation	$.
51	SYM	symbol	XY
52	TRUNC	truncated word (first part of a compound or verb prefix)	TRUNC
53	URL	URL	XY
54	VAFIN	finite auxiliary verb form	VAFIN
			VAIMP
55	VAINF	auxiliary infinitive	VAINF
56	VAPP	auxiliary past participle	VAPP
57	VMFIN	finite modal verb form	VMFIN
58	VMINF	modal infinitive	VMINF
59	VMPP	modal past participle	VMPP
60	VPREF	separated verbal prefix	PTKVZ
61	VVFIN	finite verb form	VVFIN
			VVIMP
62	VVINF	infinitive	VVINF
63	VVIZU	infinitive with incorporated “zu”	VVIZU
64	VVPP	past participle	VVPP
65	WADV	interrogative adverb	PWAV
66	WDET	interrogative determiner	PWAT
67	WINV	invariant interrogative	PIDAT
			PWAT
68	WPRO	interrogative pronoun	PWS

B.2 Additional features

No.	Tag	Description	Type
1	1P	first	person
2	2P	second	person
3	3P	third	person
4	Abk	abbreviation	additional
5	Adj	adjective	main category
6	Adpos	pre- or postposition	main category
7	Adv	adverb	main category
8	Acc	accusative	case
9	Art	article	main category
10	Attr	attributive	adjective
11	Bracket	bracket	punctuation
12	Card	cardinal	numeral
13	Circ	circumposition	main category
14	City	city	proper name type
15	Colon	colon	punctuation
16	Comma	comma	punctuation
17	Common	common	noun
18	Comp	comparative	degree
19	Conj	conjunction	main category
20	Coord	coordinating	conjunction
21	Country	country	proper name
22	Cp1	coordination, 1st part	conjunction
23	Cp2	coordination, 2nd part	conjunction
24	Dash	dash	punctuation
25	Dat	dative	case
26	Dem	demonstrative	pronoun
27	Det	determiner	main category
28	Dig	digital	numeral
29	Dots	dots	punctuation
30	Dte	date	main category
31	FMN	any gender	gender
32	Famname	family name	proper name
33	Fem	feminine	gender
34	Fract	fractal	numeral
35	Gen	genitive	case
36	Imp	imperative	mood
37	Indc	indicative	mood

No.	**Tag**	**Description**	**Type**
38	Indef	indefinite	pronoun
39	Inf	infinitive	verb
40	Init	initial	proper name
41	Invar	invariant	additional
42	Item	item	numeral
43	Itj	interjection	main category
44	Izu	infinitival *zu*	particle
45	Lang	language	noun
46	Left	left	bracket or parenthesis
47	Masc	masculine	gender
48	Math	mathematical	symbol
49	NAdj	nominalized adjective	noun
50	NGDA	any case	case
51	Neg	negative	particle
52	Neut	neutral	gender
53	Nom	nominative	case
54	Noun	common noun	main category
55	Num	spelled out numeral	numeral
56	Ord	ordinal	numeral
57	PPast	past participle	participle
58	PPres	simple present participle	participle
59	PPrzu	present participle with "zu"	participle
60	Paren	parenthesis	punctuation
61	Past	past	tense
62	Pers	personal	pronoun
63	Pl	plural	number
64	Pos	positive	degree
65	Poss	possessive	pronoun
66	Post	postposition	adposition
67	Pred	predicative or adverbial	adjective
68	Prep	preposition	adposition
69	Pres	present	tense
70	Pron	pronoun	main category
71	Prop	proper name	main category
72	Ptkl	particle	main category
73	Punct	punctuation	main category
74	Quote	quotation mark	punctuation
75	Refl	reflexive	pronoun

No.	Tag	Description	Type
76	Rel	relative	pronoun
77	Rez	reciprocal	pronoun
78	Right	right	bracket or parenthesis
79	Rom	Roman	numeral
80	Semicolon	semicolon	punctuation
81	Sent	sentence final punctuation	main category
82	Sg	singular	number
83	Slash	slash	punctuation
84	Spec	special symbol	URL
85	St	strong	declension
86	Subj	subjunctive	mood
87	Subord	subordinating	conjunction
88	Sup	with superlative	particle
89	Symbol	special symbol	main category
90	Trunc	truncated word	main category
91	Unit	physical unit	noun
92	Verb	verb	main category
93	Vorname	first name	proper name
94	Wh	interrogative or relative	pronoun, adverb
95	Wk	weak	declension

Appendix C

TüBa-D/Z treebank tags

C.1 Feature combinations for STTS tags in TüBa-D/Z

No.	STTS-Tag	Feature Combination
1	ADJA	case, number, gender
2	ADJD	–
3	ADV	–
4	APPR	case
5	APPRART	case, number, gender
6	APPO	case
7	APZR	–
8	ART	case, number, gender
9	CARD	–
10	FM	–
11	ITJ	–
12	KOUI	–
13	KOUS	–
14	KON	–
15	KOKOM	–
16	NE	case, number, gender
17	NN	case, number, gender
18	PDS	case, number, gender
19	PDAT	case, number, gender
20	PIS	case, number, gender
21	PIAT	case, number, gender
22	PIDAT	case, number, gender

No.	STTS-Tag	Feature Combination
23	PPER	case, number, gender, person
24	PPOSS	case, number, gender
25	PPOSAT	case, number, gender
26	PRELS	case, number, gender
27	PRELAT	case, number, gender
28	PRF	–
29	PWS	case, number, gender
30	PWAT	case, number, gender
31	PWAV	–
32	PROP	–
33	PTKZU	–
34	PTKNEG	–
35	PTKVZ	–
36	PTKANT	–
37	PTKA	–
38	TRUNC	number, gender
39	VVFIN	person, number, mood, tense
40	VVIMP	person, number
41	VVINF	–
42	VVIZU	–
43	VVPP	–
44	VAFIN	person, number, mood, tense
45	VAIMP	person, number
46	VAINF	–
47	VAPP	–
48	VMFIN	person, number, mood, tense
49	VMINF	–
50	VMPP	–
51	XY	–
52	$,	–
53	$.	–
54	$(	–

C.2 Set of feature values in TüBa-D/Z

Features in TüBa-D/Z	**Values**
case	n (nominative), g (genitive), d (dative), a (accusative), * (underspecified)
gender	m (masculine), f (feminine), n (neutral), * (underspecified)
number	s (singular), p (plural), * (underspecified)
mood	i (indicative), k (subjunctive)
person	1 (first), 2 (second), 3 (third)
tense	s (present), t (past)

Appendix D

TüBa-D/Z treebank labels

D.1 Node labels

Node Label	Description
	Phrase Node Labels
NCX	non-recursive noun phrase
NX	recursive noun phrase
PX	prepositional phrase
ADVX	adverbial phrase
ADJX	adjectival phrase
VXFIN	finite verb phrase
VXINF	infinite verb phrase
FX	foreign language phrase
DP	determiner phrase (e.g. *gar keine*)
EN-ADD	proper noun or named entity
	Topological Field Node Labels
LV	resumptive construction (Linksversetzung)
VF	initial field (Vorfeld)
LK	left sentence bracket (Linke (Satz-)Klammer)
MF	middle field (Mittelfeld)
VC	verb complex (Verbkomplex)
NF	final field (Nachfeld)
C	complementizer field (C-Feld)
KOORD	field for coordinating particles
PARORD	field for non-coordinating particles
FKOORD	coordination consisting of conjuncts of fields
MFE	middle field between VCE and VC

Node Label	Description
VCE	verb complex with the split finite verb of *Ersatzinfinitiv* constructions
FKONJ	conjunct consisting of more than one field
	Root Node Labels
SIMPX	simplex clause
R-SIMPX	relative clause
P-SIMPX	paratactic construction of simplex clauses
DM	discourse marker

D.2 Edge labels

Edge Labels	**Description**
Edge Labels denoting Heads and Conjuncts	
HD	head
-	non-head
KONJ	conjunct
Complement Edge Labels	
ON	nominative object
OD	dative object
OA	accusative object
OG	genitive object
OS	sentential object
OPP	prepositional object
OADVP	adverbial object
OADJP	adjectival object
PRED	predicate
OV	verbal object
FOPP	optional prepositional object
VPT	separable verb prefix
APP	apposition
Modifier Edge Labels	
MOD	ambiguous modifier
ON-MOD, OA-MOD, OD-MOD, OG-MOD, OPP-MOD, OS-MOD, PRED-MOD, FOPP-MOD, OADJP-MOD, V-MOD, MOD-MOD	modifiers modifying complements or modifiers e.g. V-MOD = modifier of the verb
Edge Labels in Split-up Coordinations	
ONK, ODK, OAK, FOPPK, OADVPK, PREDK, MODK, V-MODK	second conjunct (K) in split-up coordinations e.g. ONK = second conjunct of a nominative object (subject)
Secondary Edge Labels	
	dependency relation between:
EN	two parts of a proper noun
REFVC	two verbal objects in VC
REFMOD	two ambiguous modifiers

Edge Labels	**Description**
REFINT	a phrase internal part and its modifier
REFCONTR	control verb and its complement across clause boundaries

Appendix E

Syntactic heuristics

1. The NP is the only one in a finite clause

The heuristic will apply in two cases: either if a given NP is the only one in a finite clause, as in (118), or if a given NP is the only one that has a nominative reading in a finite clause, as in (119).

(118) Oder ist Bremerhaven nicht günstiger?
or is Bremerhaven not more cost-efficient
'Or isn't Bremerhaven more cost-efficient?'

(119) Für ein "barrierefreies Bremen" gingen deshalb gestern
for a barrier-free Bremen went therefore yesterday
mehrere hundert behinderte Menschen auf die Straße.
several hundred handicapped people into the street
'Therefore, several hundred handicapped people took to the street for a barrier-free Bremen yesterday.'

In (118), the only noun "*Bremerhaven*" will keep only one reading out of three candidates, shown in (120), after application of the heuristic. In example (119), the nominative readings for the nouns "*Bremen*" and "*Straße*" will be ruled out by prior application of a double reduction rule which requires the identity of case values between a noun and a preceding preposition. So only the noun "*Menschen*" can be the subject in this sentence. The heuristic deletes five readings out of the seven candidates in (121).

(120)	Bremerhaven	+Noun+City+Sg+Neut+Dat
	Bremerhaven	+Noun+City+Sg+Neut+Acc
	Bremerhaven	+Noun+City+Sg+Neut+Nom

(121)	Menschen	+Noun+Common+Sg+Masc+Gen
	Menschen	+Noun+Common+Sg+Masc+Dat
	Menschen	+Noun+Common+Sg+Masc+Acc
	Menschen	+Noun+Common+Pl+Masc+Nom
	Menschen	+Noun+Common+Pl+Masc+Gen
	Menschen	+Noun+Common+Pl+Masc+Dat
	Menschen	+Noun+Common+Pl+Masc+Acc

The heuristic is stated in a number of ODRs. The rules check whether there are any nominative lexemes in the right and left contexts (up to the clause boundaries) of the noun to which they apply and, if no nominative lexemes were found, delete all non-nominative readings of the noun. Possible modifiers of the noun (both pre- and post-) are not considered as competing for the subject position.

For reasons explained in Chapter 6 above, the rules have to be applied repeatedly: successive elimination of nominative readings for one or more NPs by other heuristics can make the present heuristic applicable more than once until no further disambiguation is possible.

2. A noun with feature `City` or `Country` is preceded by a preposition "*in*"

(122)	Behinderte	Menschen	veranstalteten	Protesttag	in Bremen.
	handicapped	people	organized	day of protest	in Bremen

'Handicapped people organized a day of protest in Bremen.'

As shown in (123), both the preposition "*in*" and the noun "*Bremen*" are ambiguous in case. The nominative reading of the noun will be eliminated in this context, since a noun preceded by a preposition cannot be nominative. The remaining ambiguity can be resolved due to the fact that the preposition "*in*" takes a dative complement if it refers to a city or country.[1] Presence of the features `City` and `Country` in the inventory of the Xerox morphological analyzer enables implementation of the heuristic, which otherwise would be impossible.[2]

[1]In general, the preposition "*in*" requires either dative or accusative case. If "*in*" takes an accusative NP, then "*in*" has the directional meaning of "*into*". With dative case "*in*" has locative meaning. For city and country nouns only the locative meaning of "*in*" is possible, since the directional case has to be expressed by the preposition "*nach*" for this class of NPs.

[2]For the full inventory of features see http://www.xrce.xerox.com/competencies/content-analysis/demos/doc/mor-ger-2.txt.

(123)

in	+Adpos+Prep+Acc
in	+Adpos+Prep+Dat
Bremen	+Noun+City+Sg+Neut+Dat
Bremen	+Noun+City+Sg+Neut+Acc
Bremen	+Noun+City+Sg+Neut+Nom

3. Eliminate nominative readings on ambiguous NPs if there is a non-ambiguous nominative NP in a clause

(124) Es ist wichtig, daß wir die Sängerin gut finden.
it is important that we the singer good find
'It is important that we like the singer.'

In the second clause, the pronoun "*wir*" is unambiguously nominative and the main verb is not a copula (which would require two Nom arguments), so the nominative reading of the noun phrase "*die Sängerin*" can be eliminated.[3]

(125)

wir	+Pron+Pers+1P+Pl+Fem+Nom
wir	+Pron+Pers+1P+Pl+Masc+Nom
wir	+Pron+Pers+1P+Pl+Neut+Nom

(126)

die	+Det+Art+Sg+Fem+Acc+St
die	+Det+Art+Sg+Fem+Nom+St
Sängerin	+Noun+Common+Sg+Fem+Nom
Sängerin	+Noun+Common+Sg+Fem+Acc

4. The NP is an argument of a copula verb

(127) Das Altenheim sei ein Prestigeobjekt von ihr und
the retirement home be an object of prestige of hers and
anderen.
others
'The retirement home is claimed to be an object of prestige of hers and others.'

[3]This heuristic may over-apply in some cases, for example if a nominative pronoun is followed by an appositive NP with the same case, as in "*wir müde Krieger*" ("*we tired warriors*").

A copula verb requires two nominative arguments. "*Ihr*" is disambiguated by a preceding preposition and, in its turn, allows for the application of the coordination heuristic to the indefinite adjective "*anderen*". Thus, "*Das Altenheim*" and "*ein Prestigeobjekt*" are the two arguments of the copula verb "*sei*" and receive the feature nominative, which reduces the output of DRRs to one reading.

The original set of analyses is shown in (128)–(129), the output of DRRs – in (130)–(131), and the sets of analyses for the two NPs after application of the heuristic – in (132)–(133).

(128)	Das	+Det+Art+Sg+Neut+Acc+St
	Das	+Det+Art+Sg+Neut+Nom+St
	Altenheim	+Noun+Common+Sg+Neut+Dat
	Altenheim	+Noun+Common+Sg+Neut+Acc
	Altenheim	+Noun+Common+Sg+Neut+Nom
(129)	ein	+Det+Art+Sg+Masc+Nom+Wk
	ein	+Det+Art+Sg+Neut+Acc+Wk
	ein	+Det+Art+Sg+Neut+Nom+Wk
	Prestigeobjekt	+Noun+Common+Sg+Neut+Dat
	Prestigeobjekt	+Noun+Common+Sg+Neut+Acc
	Prestigeobjekt	+Noun+Common+Sg+Neut+Nom
(130)	Das	+Det+Art+Sg+Neut+Acc+St
	Das	+Det+Art+Sg+Neut+Nom+St
	Altenheim	+Noun+Common+Sg+Neut+Acc
	Altenheim	+Noun+Common+Sg+Neut+Nom
(131)	ein	+Det+Art+Sg+Neut+Acc+Wk
	ein	+Det+Art+Sg+Neut+Nom+Wk
	Prestigeobjekt	+Noun+Common+Sg+Neut+Acc
	Prestigeobjekt	+Noun+Common+Sg+Neut+Nom
(132)	Das	+Det+Art+Sg+Neut+Nom+St
	Altenheim	+Noun+Common+Sg+Neut+Nom
(133)	ein	+Det+Art+Sg+Neut+Nom+Wk
	Prestigeobjekt	+Noun+Common+Sg+Neut+Nom

5. A nominative reading does not agree with a finite verb in number

(134) Staatsanwaltschaft muss AWO-Konten prüfen.
The prosecutor's office must AWO accounts verify
'The prosecutor's office must verify the AWO accounts.'

Both "*Staatsanwaltschaft*" and "*AWO-Konten*" have a nominative reading:

(135)	Staatsanwaltschaft	+Noun+Common+Sg+Fem+Nom
	Staatsanwaltschaft	+Noun+Common+Sg+Fem+Gen
	Staatsanwaltschaft	+Noun+Common+Sg+Fem+Dat
	Staatsanwaltschaft	+Noun+Common+Sg+Fem+Acc

(136)	AWO-Konten	+Noun+Common+Pl+Neut+Nom
	AWO-Konten	+Noun+Common+Pl+Neut+Gen
	AWO-Konten	+Noun+Common+Pl+Neut+Dat
	AWO-Konten	+Noun+Common+Pl+Neut+Acc

The finite verb, though, is unambiguously singular:

(137)	muss	+Verb+Indc+1P+Sg+Pres
	muss	+Verb+Indc+3P+Sg+Pres

There is no coordination in the sentence. Nor is "*AWO-Konten*" a part of a comparative construction, which would enable it to keep the nominative reading even though it does not agree with the finite verb in number.[4] So the nominative reading should be eliminated. Once this heuristic has been applied, the previously discussed heuristic (namely, the heuristic for the only candidate for subject) may become applicable and may lead to further disambiguation – the only noun that has a nominative reading is "*Staatsanwaltschaft*". Therefore the other readings can be eliminated.

[4]These two cases have to be guarded against by contextual constraints. Otherwise the nominative readings for conjoined NPs, as in "*Karel und Mates mögen sich sehr*" and for the plural NP in "*Zulia singt schöner als ihre Freudinnen*" would be mistakenly eliminated.

6. The NP is preceded neither by a preposition nor by another NP

(138) In einer anonymen Anzeige werden der Bremer
in an anonymous complaint were the$_{dat}$ Bremen
Staatsanwaltschaft Details über dubiose finanzielle
prosecutor's office$_{dat}$ details about dubious financial
Transaktionen mitgeteilt.
transactions disclosed

'In an anonymous complaint, the city of Bremen's prosecutor's office was given details about dubious financial transactions.'

In German, genitive is mostly used as the case of nominal modifiers and complements of prepositions. But, with a few exceptions, genitive case does not mark verb complements. This fact provides a reason for deleting a genitive reading of a noun that is neither a postmodifier of an NP nor preceded by a preposition, which is the case in the above sentence. The original set of readings (139) will be reduced by DRRs to the two analyses shown in (140). The heuristic will disambiguate the phrase completely by eliminating genitive readings.

(139)	der	+Det+Art+Pl+Fem+Gen+St
	der	+Det+Art+Pl+Masc+Gen+St
	der	+Det+Art+Pl+Neut+Gen+St
	der	+Det+Art+Sg+Masc+Nom+St
	der	+Det+Art+Sg+Fem+Dat+St
	der	+Det+Art+Sg+Fem+Gen+St
	Bremer	+Adj+Invar+Attr
	Staatsanwaltschaft	+Noun+Common+Sg+Fem+Nom
	Staatsanwaltschaft	+Noun+Common+Sg+Fem+Gen
	Staatsanwaltschaft	+Noun+Common+Sg+Fem+Dat
	Staatsanwaltschaft	+Noun+Common+Sg+Fem+Acc

(140)	der	+Det+Art+Sg+Fem+Dat+St
	der	+Det+Art+Sg+Fem+Gen+St
	Bremer	+Adj+Invar+Attr
	Staatsanwaltschaft	+Noun+Common+Sg+Fem+Gen
	Staatsanwaltschaft	+Noun+Common+Sg+Fem+Dat

The rule will cause errors in the case of a small class of verbs that require genitive complements, like "*gedenken*" ("*commemorate*"). Note, however, that the heuristic can be modified in such a way that it does not apply to nouns if such a verb is present in a clause. In the test corpus on which the grammar was evaluated the rule did not make any errors.

7. The NP is a non-initial NP in a Vorfeld position in V2 clause

(141)

Die Wahrheitsseite	dieser	Zeitung	scheint	das letzte	
the truth page	this$_{gen}$	newspaper$_{gen}$	seems	the last	
Refugium	der	Pazifisten	zu sein.		
refuge	the$_{gen}$	pacifists$_{gen}$	to be		

'The "truth page" of this newspaper seems to be the last refuge of the pacifists.'

The Vorfeld is the first constituent in a verb-second clause. This position can be occupied by only one element or phrase. Thus, if it contains more than one phrase, all but the first phrase are modifiers of the preceding elements. Since in German a postmodifying NP with no preceding preposition has to be genitive, readings with all other case values can be eliminated.

The original analyses of the NP "*dieser Zeitung*" are as in (142).

(142)

dieser	+Det+Dem+Sg+Fem+Dat+St
dieser	+Det+Dem+Sg+Fem+Gen+St
dieser	+Det+Dem+Sg+Masc+Nom+St
dieser	+Det+Dem+Pl+Fem+Gen+St
dieser	+Det+Dem+Pl+Masc+Gen+St
dieser	+Det+Dem+Pl+Neut+Gen+St
Zeitung	+Noun+Common+Sg+Fem+Nom
Zeitung	+Noun+Common+Sg+Fem+Gen
Zeitung	+Noun+Common+Sg+Fem+Dat
Zeitung	+Noun+Common+Sg+Fem+Acc

After the application of DRRs only two readings will be left:

(143)

dieser	+Det+Dem+Sg+Fem+Dat+St
dieser	+Det+Dem+Sg+Fem+Gen+St
Zeitung	+Noun+Common+Sg+Fem+Gen
Zeitung	+Noun+Common+Sg+Fem+Dat

The heuristic will further eliminate dative readings, thus resulting in an unambiguous output.

8. The NP is a complement of a zu-infinitive

(144) Klarer Regelverstoß und Grund genug, die ehemalige Siegerin zu
clear rule violation and reason enough, the former winner to
disqualifizieren.
disqualify
'A clear rule violation and sufficient grounds for disqualifying the former winner.'

A zu-infinitive is a non-finite clause, so that the nominative reading can be eliminated. The set of original analyses for the NP, as shown in (145), will decrease to two analyses (146) after the application of DRRs. The heuristic for zu-infinitive then helps to disambiguate the phrase completely.

(145)	die	+Det+Art+Pl+Fem+Acc+St
	die	+Det+Art+Pl+Masc+Acc+St
	die	+Det+Art+Pl+Neut+Acc+St
	die	+Det+Art+Pl+Fem+Nom+St
	die	+Det+Art+Pl+Masc+Nom+St
	die	+Det+Art+Pl+Neut+Nom+St
	die	+Det+Art+Sg+Fem+Acc+St
	die	+Det+Art+Sg+Fem+Nom+St
	ehemalige	+Adj+Pos+Pl+Fem+Acc+St
	ehemalige	+Adj+Pos+Pl+Masc+Acc+St
	ehemalige	+Adj+Pos+Pl+Neut+Acc+St
	ehemalige	+Adj+Pos+Pl+Fem+Nom+St
	ehemalige	+Adj+Pos+Pl+Masc+Nom+St
	ehemalige	+Adj+Pos+Pl+Neut+Nom+St
	ehemalige	+Adj+Pos+Sg+Fem+Nom+Wk
	ehemalige	+Adj+Pos+Sg+Masc+Nom+Wk
	ehemalige	+Adj+Pos+Sg+Neut+Nom+Wk
	ehemalige	+Adj+Pos+Sg+Fem+Acc+St
	ehemalige	+Adj+Pos+Sg+Fem+Acc+Wk
	ehemalige	+Adj+Pos+Sg+Fem+Nom+St
	ehemalige	+Adj+Pos+Sg+Neut+Acc+Wk
	Siegerin	+Noun+Common+Sg+Fem+Nom
	Siegerin	+Noun+Common+Sg+Fem+Gen
	Siegerin	+Noun+Common+Sg+Fem+Dat
	Siegerin	+Noun+Common+Sg+Fem+Acc

(146)	die	+Det+Art+Sg+Fem+Acc+St
	die	+Det+Art+Sg+Fem+Nom+St
	ehemalige	+Adj+Pos+Sg+Fem+Nom+Wk
	ehemalige	+Adj+Pos+Sg+Fem+Acc+Wk
	Siegerin	+Noun+Common+Sg+Fem+Nom
	Siegerin	+Noun+Common+Sg+Fem+Acc

Appendix F

Derivation of the formula for tagging with Markov models

When a Markov model is applied to tagging, the states correspond to tags and the output symbols correspond to words. The task is to find the most likely sequence of tags $t_{1,k}$ for a string $w_{1,k}$, i.e. a tag sequence that maximizes the conditional probability $P(t_{1,k} \mid w_{1,k})$:

$$\underset{t_{1,k}}{argmax} P(t_{1,k} \mid w_{1,k}) = \underset{t_{1,k}}{argmax} \frac{P(t_{1,k})P(w_{1,k} \mid t_{1,k})}{P(w_{1,k})}$$

$P(w_{1,k})$ is constant for all $t_{1,k}$, so it is sufficient to find

$$\underset{t_{1,k}}{argmax} P(t_{1,k})P(w_{1,k} \mid t_{1,k})$$

1. $$\begin{aligned} P(t_{1,k}) &= P(t_1)P(t_2 \mid t_1)P(t_3 \mid t_1 t_2)\ldots P(t_k \mid t_1 \ldots t_{k-1}) \\ &= P(t_1)\prod_{i=2}^{k} P(t_i \mid t_1 \ldots t_{i-1}) \\ &= P(t_1)\prod_{i=2}^{k} P(t_i \mid t_{i-1}) \end{aligned}$$

   The last simplification of the equation is received by making a Markov assumption $P(t_i \mid t_1 \ldots t_{i-1}) = P(t_i \mid t_{i-1})$.

2. $$\begin{aligned} P(w_{1,k} \mid t_{1,k}) &= P(w_1 \mid t_{1,k})P(w_2 \mid t_{1,k}, w_1)\ldots P(w_k \mid t_{1,k}, w_{1,k-1}) \\ &= P(w_1 \mid t_{1,k})\prod_{i=2}^{k} P(w_i \mid t_{1,k}, w_{1,i-1}) \end{aligned}$$

$$= P(w_1 \mid t_{1,k}) \prod_{i=2}^{k} P(w_i \mid t_i)$$

The last simplification of the equation is received by making a Markov assumption $P(w_i \mid t_{1,k}, w_{1,i-1}) = P(w_i \mid t_i)$.

3. $P(t_{1,k})P(w_{1,k} \mid t_{1,k}) = P(t_1)P(w_1 \mid t_{1,k}) \prod_{i=2}^{k} P(t_i \mid t_{i-1})P(w_i \mid t_i)$

   Introducing the initial state t_0,

$$= \prod_{i=1}^{k} P(t_i \mid t_{i-1})P(w_i \mid t_i)$$

Thus,

$$\underset{t_{1,k}}{argmax} P(t_{1,k})P(w_{1,k} \mid t_{1,k}) = \underset{t_{1,k}}{argmax} \prod_{i=1}^{k} P(t_1 \mid t_{i-1})P(w_i \mid t_i)$$

The formula describes a first order Markov model, or a *bigram* model. By making a more general Markov assumption:

$$P(t_i \mid t_1 \dots t_{i-1}) = P(t_i \mid t_{i-n+1} \dots t_{i-1})$$

a general formula for $(n-1)^{th} order$ Markov model, or an *n-gram* model can be received:

$$\underset{t_{1,k}}{argmax} P(t_{1,k})P(w_{1,k} \mid t_{1,k}) = \underset{t_{1,k}}{argmax} \prod_{i=1}^{k} P(t_1 \mid t_{i-n+1} t_{i-1})P(w_i \mid t_i)$$

Appendix G

Algorithms for reducing a tagset

G.1 T-Tagging approach of Tufiş (2000)

Notation

MSD	Morpho-syntactic description codes: tags of the initial large tagset.
MSD-lexicon	Lexicon which provides MSDs for each token.
C-tag	Reduced tagset: a tagset used for an intermediate tagging.

The algorithm

```
1. extract all ambiguity classes from the MSD-lexicon

2. normalize all MSD ambiguity classes (i.e. mark all the attributes that
   take all possible values as irrelevant for the ambiguity class in case)

3. **for each** ambiguity class $AC_i$
     preserve only intra-categorial ambiguities: $\mathrm{ICA}_i$

4. **for each** $\mathrm{ICA}_i$ **repeat**
     **for each** $\mathrm{MSD}_{ij}$ **repeat**
       **for each** attribute $\mathrm{A}_k$ in $\mathrm{MSD}_{ij}$ **repeat**
         **if** eliminating $\mathrm{A}_k$ does not reduce the cardinality of any of ICAs
           **then** remove $\mathrm{A}_k$ from all tags in all ICAs
```

```
        else if eliminating $A_k$ does not reduce the cardinality of more
                than 10% of ICAs
          then mark $A_k$ as removable

5. for all $A_k$ marked as removable
        compute the maximal set of attributes that minimally reduces
        the cardinality of all ICAs (not unique solution)

6. for each Ctag-set obtained in step 5 evaluate the performance
```

An example

MSD-AC$_i$ = (Afpms-n Ncms-n Vmp–sm) ==> no ICAs.
MSD-AC$_j$ = (Ncfp-n Ncfson Vmis3s Vmm-2s Vmnp) ==> 2 ICAs:
(Ncfp-n Ncfson) and (Vmis3s Vmm-2s Vmnp).

Eventually, the algorithm will lead to:

Ctag-AC$_i$ = (ASN NSN VP)
Ctag-AC$_j$ = (NPN NSPN V3 V2 VN)

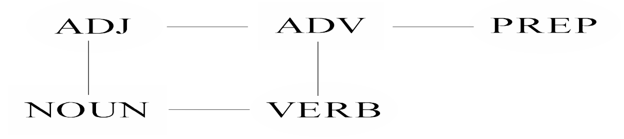
G.2 BUTD approach of Dienes and Oravecz (2000)

Algorithm

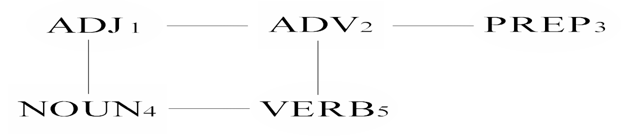
1. Establish graph G whose nodes are the tags of the initial tagset. Connect two nodes if they occur in the same ambiguity class.
2. Order the nodes of the graph in any way.
3. Establish a set of classes (= resulting tagset). Order the classes in any way.
4. For each i = 1,2,... assign the i-th node to a class with the smallest available number. Make this class unavailable for all neighboring nodes.

An example

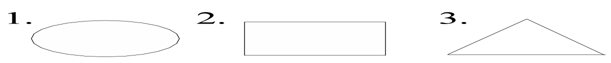
1.



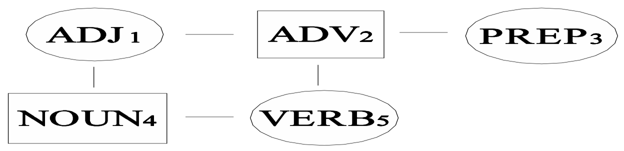
2.



3.



4.



Resulting classes: 1) ADJ, PREP, VERB; 2) ADV, NOUN.

Bibliography

Aït-Mokhtar, Salah, Jean-Pierre Chanod and Claude Roux (2002), 'Robustness beyond shallowness: Incremental deep parsing', *Journal of Natural Language Engineering* **8**(3), 121–144.

Anderson, John (1971), *The Grammar of Case: Towards a Localist Theory*, Cambridge University Press, Cambridge, England.

Anderson, John (1979), 'Syntax and the single mother', *Journal of Linguistics* **15**.

Baum, LE, T. Petrie, G. Soules and N. Weiss (1970), 'A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains', *Annals of Mathematical Statistics* **14**(164-171).

Bech, C. (1955-57), *Studien über das deutsche Verbum infinitum.*, Max Niemeyer, Tübingen, Germany.

Benello, Julian, Andrew W. Mackie and James A. Anderson (1989), 'Syntactic category disambiguation with neural networks', *Computer Speech and Language* **3**, 203–217.

Bloomfield, Leonard (1933), *Language*, New York: Holt, Rinehart and Winston.

Brants, Thorsten (1998), *TnT–A Statistical Part-of-Speech Tagger*, Universität des Saarlandes, Computational Linguistics, Saarbrücken, Germany.
URL: *http://www.coli.uni-sb.de/~thorsten/tnt/*

Brants, Thorsten (2000), TnT–a statistical part-of-speech tagger, *in* 'Proceedings of the 6th Conference on Applied Natural Language Processing, ANLP-2000', Seattle, WA.
URL: *http://xxx.lanl.gov/ps/cs.CL/0003055*

Bresnan, Joan, ed. (1982), *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, MA.

Brill, Eric (1992), A simple rule-based part of speech tagger, *in* 'Proceedings of the DARPA Speech and Natural Language Workshop', Harriman, NY, pp. 112–116.

Brill, Eric (1995*a*), 'Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging', *Computational Linguistics* **24**(1), 543–565.

Brill, Eric (1995*b*), Unsupervised learning of disambiguation rules for part of speech tagging, *in* 'Proceedings of the 3rd Workshop on Very Large Corpora (WVLC 3)', pp. 1–13.

Bröker, Norbert (1998), Separating surface order and syntactic relations in a dependency grammar, *in* 'Proceedings of COLING-ACL'98', pp. 174–180.

Bröker, Norbert, Udo Hahn and Susanne Schacht (1994), Concurrent lexicalized dependency parsing: the ParseTalk model, *in* 'Proceedings of COLING'94', Kyoto, Japan.

Brown, Peter F., Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai and Robert L. Mercer (1992), 'Class based n gram models of natural language', *Computational Linguistics* **18**(4), 467–469.

Cardie, Claire (1993), Using decision trees to improve case-based learning, *in* 'Proceedings of the Tenth International Conference on Machine Learning', Morgan Kaufmann, Amherst, MA, pp. 25–32.

Carroll, Glenn and Eugene Charniak (1992), Two experiments on learning probabilistic dependency grammars from corpora, *in* 'Proceedings from AAAI-92 Workshop Program', AAAI Press, San Jose, CA.

Chanod, Jean-Pierre and Pasi Tapanainen (1995), Tagging French - comparing a statistical and a constraint-based method, *in* 'Proceedings of the 7th Conference of the EACL', Dublin, Ireland, pp. 149–156.

Charniak, Eugene (2000), A maximum-entropy-inspired parser, *in* 'Proceedings of ANLP/NAACL'2000', Seattle, WA, pp. 132–139.

Chomsky, Noam (1981), *Lectures on Government and Binding*, Foris Publications, Dordrecht.

Chung, Hoojung (2004), Statistical Korean Dependency Parsing Model based on the Surface Contextual Information, PhD thesis, Korea University, Seoul.

Church, Kenneth W. (1988), A stochastic parts program and noun phrase parser for unrestricted text, *in* 'Proceedings of the Second ACL Conference on Applied Natural Language Processing', Austin, TX, pp. 136–143.

Collins, Michael (1999), Head-Driven Statistical Models for Natural Language Parsing, PhD thesis, University of Pennsylvania.

Collins, Michael John (1996), A new statistical parser based on bigram lexical dependencies, *in* 'Proceedings of ACL 96', Santa Cruz, CA.
URL: *http://xxx.lanl.gov/ps/cmp-lg/9605012*

Cutting, Doug, Julian Kupiec, Jan Pedersen and Penelope Sibun (1992), A practical part-of-speech tagger, *in* 'Proceedings of the Third ACL Conference on Applied Natural Language Processing', Trento, Italy, pp. 133–140.

Daelemans, Walter, Jakub Zavrel and Peter Berck (1996), Part-of-speech tagging of Dutch with MBT, a memory-based tagger generator, *in* 'Proceedings Informatiewetenschap', pp. 33–40.

Daelemans, Walter, Jakub Zavrel, Peter Berck and Steven Gillis (1996), MBT: A memory-based part of speech tagger-generator, *in* E.Ejerhed and I.Dagan, eds., 'Proceedings of the Fourth Workshop on Very Large Corpora', Copenhagen, Denmark, pp. 14–27.

Darroch, J. N. and D. Ratcliff (1972), 'Generalized iterative scaling for log-linear models', *The Annals of Mathematical Statistics* **43**(5), 1470–1480.

DeRose, Steven J. (1988), 'Grammatical category disambiguation by statistical optimization', *Computational Linguistics* **14**(1), 31–39.

Dienes, Péter and Csaba Oravecz (2000), Bottom-up tagset design from maximally reduced tagset, *in* 'Proceedings of the Workshop on Linguistically Interpreted Corpora LINC-2000', Luxembourg, pp. 42–47.

Drach, Erich (1937), *Grundgedanken der Deutschen Satzlehre*, Diesterweg, Frankfurt/M.

Duchier, Denys (1999), Axiomatizing dependency parsing using set constraints, *in* 'Proceedings of the Sixth Meeting on Mathematics of Language', Orlando, Florida, pp. 115–126.

Duchier, Denys and Ralph Debusmann (2001), Topological dependency trees: A constraint-based account of linear precedence, *in* 'Proceedings of ACL'2001', Toulouse, France.

Eisner, Jason (1996), Three new probabilistic models for dependency parsing: An exploration, *in* 'Proceedings of COLING'96', Copenhagen.

Elworthy, David (1999), Finite-state parser with dependency structure output, *in* 'Proceedings of the 6th International Workshop on Parsing Technology', Trento, Italy.

Engel, Ulrich (1972), 'Bemerkunger der Dependenzgrammatik'. Neue Grammatiktheorien und ihre Anwendung auf das heutige Deutsch, 20.

Erdmann, Oskar (1886), *Grundzüge der deutschen Syntax nach ihrer geschichtlichen Entwicklung dargestellt.*, Verlag der Cotta'schen Buchhandlung, Stuttgart, Germany. Erste Abteilung.

Erjavec, Tomaž, Sašo Džeroski and Jakub Zavrel (1999), Morphosyntactic tagging of Slovene: Evaluating PoS taggers and tagsets, Technical report, Dept. for Intelligent Systems, Jozef Stefan Institute, Ljubljana, Slovenia.

Fillmore, Charles (1968), The case for case, *in* E.Bach and R.Harms, eds., 'Universals in Linguistic Theory', New York.

Fraser, Norman M. (1994), Dependency grammar, *in* R.Asher, ed., 'Encyclopedia of Language and Linguistics', Oxford: Pergamon Press, pp. 860–864.

Gaifman, Haim (1965), 'Dependency systems and phrase-structure systems', *Information and Control* **8**(3), 304–337.

Gazdar, Gerald, Ewan Klein, GK Pullum and Ivan Sag. (1985), *Generalised Phrase Structure Grammar, Oxford*, Basil Blackwell,, Oxford.

Gerdes, Kim and Sylvain Kahane (2001), Word order in German: a formal dependency grammar using a topological hierarchy, *in* 'Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics', Toulouse, Franc.

Germann, Ulrich (1999), A deterministic dependency parser for japanese, *in* 'Proceedings of the MT Summit VII: MT in the Great Translation Era.', Association for Machine Translation., Singapore, pp. 547–555.

Greene, Barbara B. and Gerald M. Rubin (1971), Automatic grammatical tagging of english, Technical report, Brown University, Providence, RI.

Hahn, Udo, Norbert Bröker and Peter Neuhaus (2000), Let's ParseTalk: Message-passing protocols for object-oriented parsing, *in* H.Bunt and A.Nijholt, eds., 'Advances in Probabilistic and other Parsing Technologies', Kluwer, Dordrecht, Boston, London, pp. 177–201.

Hajič, Jan and Barbora Hladka (1997), Probabilistic and rule-based tagger of an inflective language - a comparison, *in* 'Proceedings of ANLP'97', Washington, D.C., pp. 111–118.

Hajič, Jan and Barbora Hladka (1998), Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset., *in* 'Proceedings of the 17th international conference on Computational linguistics', Montreal, Quebec, Canada.

Hajič, Jan, Pavel Krbec, Pavel Květoň, Karel Oliva and Vladimír Petkevič (2001), Serial combination of rules and statistics: A case study in Czech tagging, *in* 'Proceedings ACL'2001', Toulouse, France, pp. 260–267.

Harris, Zellig (1962), *String Analysis of Language Structure*, Mouton and Co., The Hague.

Hays, David G. (1964), 'Dependency theory: A formalism and some observations', *Language* **40**(4), 511–525.

Heringer, Hans Jürgen (1970), Einige Ergebnisse und Probleme der Dependenzgrammatik. Der Deutschunterricht.

Herling, Simon Heinrich Adolf (1821), Über die Topik der deutschen Sprache, *in* 'Abhandlungen des frankfurterischen Gelehrtenvereins für deutsche Sprache', Frankfurt/M., pp. 296–362, 394. Drittes Stück.

Hinrichs, Erhard W., Julia Bartels, Yasuhiro Kawata, Valia Kordoni and Heike Telljohann (2000), The Verbmobil treebanks, *in* '5. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2000)', Ilmenau, Germany, pp. 107–112.

Hinrichs, Erhard W. and Julia Trushkina (2002), 'Forging agreement: Morphological disambiguation of noun phrases', *Journal of Language and Computation* p. to appear.

Höhle, Tilman (1985), Der Begriff "Mittelfeld", Anmerkungen über die Theorie der topologischen Felder, *in* 'Akten des Siebten Internationalen Germanistenkongresses 1985', Göttingen, pp. 329–340.

Hudson, Richard (1976), *Arguments for a Nontransformational Grammar*, University of Chicago Press, Chicago.

Hudson, Richard (1984), *Word grammar*, Oxford: Blackwell.

Hudson, Richard (1990), *English Word Grammar*, Oxford, Basil Blackwell.

Hudson, Richard (1993), Recent developments in dependency theory, *in* J.Jacobs, W.Sternefeld, T.Vennemann and A.von Stechow, eds., 'Syntax: An International Handbook of Contemporary Research', De Gruyter, pp. 329–338.

Hudson, Richard (2000), 'Dependency grammar', Course Materials for the European Summer School in Logic, Language and Information.
URL: *http://www.phon.ucl.ac.uk/home/dick/esslli.htm*

Hudson, Richard (2003), The psychological reality of syntactic dependency relations, *in* S.Kahane and A.Nasr, eds., 'Proceedings of the First International Conference on Meaning-Text Theory', Paris:Ecole Normale Supérieure, pp. 181–192.

Hurskainen, Arvi (1996), Disambiguation of morphological analysis in Bantu languages, *in* 'Proceedings of COLING'96', Copenhagen, Denmark, pp. 568–573.

Jackendoff, Ray (1977), *X-bar syntax: A study of phrase structure*, MIT Press, Cambridge, MA.

Järvinen, Timo and Pasi Tapanainen (1998), Towards an implementable dependency grammar, *in* S.Kahane and A.Polguere, eds., 'Proceedings of the Workshop "Processing of Dependency-Based Grammars"', Quebec, Canada, pp. 1–10.

Jelinek, Frederick (1985), Markov source modeling of text generation, *in* J. K.Skwirzinski, ed., 'The Impact of Processing Techniques on Communications', Martinus Nijhoff, pp. 569–598.

Johnson, Mark (1998), The effect of alternative tree representations on tree bank grammars, *in* D. M. W.Powers, ed., 'Proceedings of NeMLaP3/CoNLL98', Sydney, Australia, pp. 39–48.

Karlsson, Fred (1990), Constraint grammar as a framework for parsing running text, *in* 'Proceedings of COLING'90'.

Karlsson, Fred, Atro Voutilainen, Juha Heikkilä and Arto Anttila, eds. (1995), *Constraint Grammar : a language-independent system for parsing unrestricted text*, Mouton de Gruyter, Berlin and New York.

Karttunen, Lauri, Ronald M. Kaplan and Annie Zaenen (1992), Two-level morphology with composition, *in* 'Proceedings of the International Conference on Computational Linguistics (COLING'92)', Nantes, France, pp. 141–148.

Kempe, André (1993), 'A stochastic tagger and an analysis of tagging errors', Internal paper, Institute for Computational Linguistics, University of Stuttgart.

Klein, Sheldon and Robert F. Simmons (1963), 'A computational approach to grammatical coding of English words', *Journal of the ACM* **10**, 334–347.

Kruijff, Geert-Jan M. and Denys Duchier (2002), Formal and computational aspects of dependency grammar, *in* 'Course Materials for the European Summer School in Logic, Language and Information', Trento, Italy.
URL: *http://www.coli.uni-sb.de/~gj/Lectures/DG.ESSLLI/*

Kübler, Sandra and Heike Telljohann (2002), Towards a dependency-based evaluation for partial parsing, *in* 'Proceedings of the LREC-Workshop Beyond PARSEVAL–Towards Improved Evaluation Measures for Parsing Systems', LREC 2002, Las Palmas, Gran Canaria, pp. 9–16.

Kunze, Jürgen (1975), *Abhängigkeitsgrammatik*, Akademie-Verlag, Berlin.

Kupiec, Julian (1989), Augmenting a hidden markov model for phrase-dependent word tagging, *in* 'In DARPA Speech and Natural Language Workshop', Morgan Kaufmann, Cape Cod, MA, pp. 92–98.

Lezius, Wolfgang, Reinhard Rapp and Manfred Wettler (1996), A morphology-system and part-of-speech tagger for German, *in* D.Gibbon, ed., 'Natural Language Processing and Speech Technology. Results of the 3rd KONVENS Conference', Mouton de Gruyter, Bielefeld, Germany.

Lezius, Wolfgang, Reinhard Rapp and Manfred Wettler (1998), A freely available morphological analyzer, disambiguator and context sensitive lemmatizer for German, *in* 'Proceedings of COLING-ACL'98'.

Lin, Dekang (1995), A dependency-based method for evaluating broad-coverage parsers, *in* 'Proceedings of IJCAI'95', Montreal, Canada, pp. 1420–1425.

Ma, Qing and Hitoshi Isahara (1997), Part-of-Speech tagging of Thai corpus with the logically combined neural networks, *in* 'Proceedings of the Natural Language Processing Pacific Rim Symposium', pp. 537–540.

Magerman, David (1995), Statistical decision-tree models for parsing, *in* 'Proceedings of the 33rd Annual Meeting of the ACL', Cambridge, MA, pp. 276–283.

Manning, Christopher D. and Hinrich Schütze (1999), *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, MA.

Marcus, Mitchell, Beatrice Santorini and Mary Ann Marcinkiewicz (1993), 'Building a large annotated corpus of English: The Penn Treebank', *Computational Linguistics* **19**(2), 313–330.

Marques, Nuno C. and Gabriel Pereira Lopes (1996), Using neural nets for Portuguese Part-of-Speech tagging, *in* 'Proceedings of the Fifth International Conference on the Cognitive Science of Natural Language Processing', Dublin, Ireland.

Márquez, Lluìs (1999), Part-of-Speech Tagging: A Machine-Learning Approach based on Decision Trees, PhD thesis, Software Department, Technical University of Catalonia.

Maruyama, Hiroshi (1990), Constraint dependency grammar, Technical Report RT0044, IBM, Tokyo, Japan.

Matsubara, Shigeki, Takahisa Murase, Nobuo Kawaguchi and Yasuyoshi Inagaki (2002), Stochastic dependency parsing of spontaneous japanese spoken language, *in* 'Proceedings of 17th International Conference on Computational Linguistics (COLING-2002)', Vol. 1, Taipei, Taiwan, pp. 640–645.

Matthews, Peter (1981), *Syntax*, Cambridge: University Press.

Megyesi, Beáta (2001), Comparing data-driven learning algorithms for pos tagging of swedish, *in* 'Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'2001)', Pittsburgh, PA, USA, pp. 151–158.

Mel'čuk, Igor A. (1988), *Dependency Syntax: Theory and Practise*, SUNY Series in Linguistics, State University of New York Press, Albany, NY.

Merialdo, Bernard (1994), 'Tagging English with a probabilistic model', *Computational Linguistics* **20**(2), 155–171.

Nivre, Joakim, Johan Hall and Jens Nilsson (2004), Memory-based dependency parsing, *in* 'Proceedings of CoNLL-2004', Boston, MA, USA, pp. 49–56.

Oflazer, Kemal (1999), Dependency parsing with an extended finite state aproach, *in* 'Proceedings of ACL 1999'.

Oflazer, Kemal and Gökhan Tür (1996), Combining hand-crafted rules and unsupervised learning in constraint-based morphological disambiguation, *in* 'Proceedings of the ACL-SIGDAT'96', Philadelphia, PA, USA.

Oflazer, Kemal and Gökhan Tür (1997), Morphological disambiguation by voting constraints, *in* 'Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL 1997)', pp. 122–129.

Oflazer, Kemal and Ilker Kuruöz (1994), Tagging and morphological disambiguation of Turkish text, *in* 'Proceedings of ANLP'94', Las Cruces, New Mexico, USA.

Orphanos, Giorgos, Dimitris Kalles, Thanasis Papagelis and Dimitris Christodoulakis (1999), Decision trees and NLP: A case study in POS tagging, *in* 'Proceedings of the ECCAI Advanced Course on Artificial Intelligence', Chania, Greece.

Pérez-Ortiz, Juan Antonio and Mikel L. Forcada (2001), Part-of-speech tagging with recurrent neural networks, *in* 'Proceedings of the International Joint Conference on Neural Networks', pp. 1588–1592.

Petkevič, Vladimir (2001), Grammatical agreement and automatic morphological disambiguation of inflectional languages, *in* V.Matoušek, P.Mautner, R.Mauček and K.Taušer, eds., 'Proceedings of the International Conference on Text Speech and Dialogue (TSD 2001)', Springer Verlag, Berlin, 2001, pp. 47–53.

Plaehn, Oliver (1998), *Annotate Bedienungsanleitung*, Universität des Saarlandes, Sonderforschungsbereich 378, Projekt C3, Saarbrücken, Germany.

Pollard, Carl and Ivan Sag (1994), *Head-Driven Phrase Structure Grammar*, Studies in Contemporary Linguistics, University of Chicago Press, Chicago, IL.

Quinlan, John Ross (1983), Learning efficient classification procedured and thier application to chess and games., *in* J. C.R. Michalski and T.Mitchell, eds., 'Machine Learning: An artificial intelligence approach', Morgan Kaufmann, San Mateo, CA, pp. 463–482.

Quinlan, John Ross (1986), 'Induction of decision trees', *Machine Learning* **1**, 81–206.

Quinlan, John Ross (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA.

Ratnaparkhi, Adwait (1996), A maximum entropy model for part-of-speech tagging, *in* 'Proceedings of the First Conference on Empirical Methods in Computational Linguistics (EMNLP'96)', pp. 133–142.

Reis, Marga (1982), Zum Subjektbegriff des Deutschen, *in* W.Abraham, ed., 'Satzglieder im Deutschen. Vorschlge zur syntaktischen, semantischen und pragmatischen Fundierung.', Tübingen: Narr, pp. 171–210.

Robinson, Jane (1970), 'Dependency structures and transformational rules', *Language* **46**, 259–285.

Samuelsson, Christer (1993), Morphological tagging based entirely on bayesian inference, *in* 'Proceedings of the 9th Nordic Conference on Computational Linguistics', Stockholm, Sweden.

Samuelsson, Christer and Atro Voutilainen (1997), Comparing a linguistic and a stochastic tagger, *in* 'Proceedings of the 35th Annual Meeting of the ACL and 8th Conference of the European Chapter of the ACL'.

Schacht, Susanne, Udo Hahn and Norbert Bröker (1994), Concurrent lexicalized dependency parsing: a behavioral view on ParseTalk events, *in* 'Proceedings of COLING'94', Kyoto, Japan, pp. 379–385.

Schiller, Anne, Simone Teufel and Christine Thielen (1995), Guidelines für das Tagging deutscher Textkorpora mit STTS, Technical report, Universität Stuttgart and Universität Tübingen.
URL: *http://www.sfs.nphil.uni-tuebingen.de/Elwis/stts/stts.html*

Schmid, Helmut (1994*a*), Part-of-speech tagging with neural networks, *in* 'Proceedings of the International Conference on Computational Linguistics', pp. 172–176.

Schmid, Helmut (1994*b*), Probabilistic part-of-speech tagging using decision trees, *in* 'Proceedings of the International Conference on New Methods in Language Processing'.

Schmid, Helmut (1995), Improvements in part-of-speech tagging with an application to German, *in* 'Proceedings of the 14th International Conference on Computational Linguistics', Kyoto, Japan, pp. 172–176.

Schmid, Helmut (2000), Lopar: Design and implementation, Technical Report 149, IMS Stuttgart. Arbeitspapiere des Sonderforschungsbereiches 340.

Schmid, Helmut and André Kempe (1996), Tagging von korpora mit hmm, entscheidungsbäumen und neuronalen netzen, *in* E.Feldweg, H.; Hinrichs, ed., 'Lexicographica', Vol. 73, Niemeyer Verlag, Tübingen, Germany, pp. 231–244.

Schröder, Ingo (2002), Natural Language Parsing with Graded Constraints, PhD thesis, Department of Computer Science. University of Hamburg, Hamburg, Germany.

Schröder, Ingo, Wolfgang Menzel, Kilian Foth and Michael Schulz (2000), 'Modeling dependency grammar with restricted constraints', *Traitement Automatique des Langues* **1**, 97–126.

Sgall, Petr, Eva Hajičová and Jarmila Panevová (1986), *The meaning of the sentence in its semantic and pragmatic aspects*, D. Reidel, Dordrecht.

Skut, Wojciech, Brigitte Krenn, Thorsten Brants and Hans Uszkoreit (1997), An annotation scheme for free word order languages, *in* 'Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP)', Washington, D.C.
URL: *http://xxx.lanl.gov/ps/cmp-lg/9702004*

Starosta, Stanley (1988), *The Case for Lexicase: An Outline of Lexicase Grammatical Theory*, Pinter Publisher, London.

Stegmann, Rosmary, Heike Telljohann and Erhard W. Hinrichs (2000), Stylebook for the German Treebank in VERBMOBIL, Technical Report 239, Verbmobil.

Steiner, Petra (1995), Anforderung und Probleme beim Taggen deutscher Zeitungstexte, *in* Feldweg and Hinrichs, eds., 'Lexicon und Text', Niemeyer, Tübingen.

Tapanainen, Pasi and Atro Voutilainen (1994), Tagging accurately – Don't guess if you know, *in* 'Proceedings of the 4th Conference on Applied Natural Language Processing, ANLP-1994', Stuttgart, Germany, pp. 47–52.

Tapanainen, Pasi and Timo Järvinen (1997), A non-projective dependency parser, *in* 'Proceedings of the 5th Conference on Applied Natural Language Processing', Washington, D.C.

taz (1999). Die Tageszeitung (CD-ROM). September 1986 – May 1999.
URL: *http://www.taz.de*

Telljohann, Heike, Erhard W. Hinrichs and Sandra Kübler (2003), *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*, Seminar für Sprachwissenschaft, Universität Tübingen, Tübingen, Germany.
URL: *http://www.sfs.uni-tuebingen.de/en_tuebadz.shtml*

Tesnière, Lucien (1953), *Esquisse d'une syntaxe structurale*, Klincksieck, Paris.

Tesnière, Lucien (1959), *Eléments de la syntaxe structurale*, Klincksieck, Paris.

Tufiş, Dan (2000), Using a large set of EAGLES-compliant morpho-syntactic descriptors as a tagset for probabilistic tagging, *in* 'Proceedings of the Second International Conference on Language Resources and Evaluation LREC'2000', Athens, Greece, pp. 1105–1112.

Tufiş, Dan, Péter Dienes, Csaba Oravecz and Tamás Váradi (2000), Principled hidden tagset design for Tiered Tagging of Hungarian, *in* 'International Conference on Language Resources and Evaluation LREC'2000', Athens, Greece, pp. 1421–1426.

Viterbi, A. (1967), 'Error bounds for convolutional codes and an asymptotically optimum decoding algorithm', *IEEE Transactions on Information Theory* **IT-13**, 260–269.

Volk, Martin and Gerold Schneider (1998), Comparing a statistical and a rule-based tagger for German, *in* 'Proceedings of KONVENS'98', Bonn.

Voutilainen, Atro (1995*a*), Morphological disambiguation, *in* F.Karlsson, A.Voutilainen, J.Heikkilä and A.Anttila, eds., 'Constraint Grammar', Mouton de Gruyter, Berlin, pp. 165–285.

Voutilainen, Atro (1995*b*), A syntax-based part-of-speech analyser, *in* 'Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics', European Chapter of the Association for Computational Linguistics, Dublin, Ireland.

Voutilainen, Atro (1997), EngCG tagger, Version 2, *in* T.Brondsted and I.Lytje, eds., 'Sprog og Multimedier', Aalborg Universitetsforlag, Aalborg.

White, Allan P. and Wei Zhong Liu (1994), 'Bias in information-based measures in decision tree induction', *Machine Learning* **15**(3), 321–329.

Wothke, Klaus, Ilona Weck-Ulm, Johannes Heinecke, Oliver Mertineit and Thomas Pachunke (1993), Statistically based automatic tagging of German text corpora with Parts-of-Speech, Technical Report 75.93.02, IBM Wissenschaftliches Zentrum, Heidelberg.

Yamada, Hiroyasu and Yuji Matsumoto (2003), Statistical dependency analysis with support vector machines, *in* 'Proceedings of the 8th International Workshop on Parsing Technologies (IWPT'03)', pp. 195–206.

Zwicky, Arnold (1986), 'German agreement in GPSG', *Linguistics* **24**, 957–990.