A rule-based language model for speech recognition

Author(s):
Kaufmann, Tobias

Publication Date:
2009

Permanent Link:
https://doi.org/10.3929/ethz-a-006035932

Rights / License:
In Copyright - Non-Commercial Use Permitted
A Rule-based Language Model for Speech Recognition

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZURICH

for the degree of
DOCTOR OF SCIENCES

presented by
TOBIAS KAUFMANN
Dipl. Informatik-Ing. ETH
born July 16, 1978
citizen of Meinisberg (BE), Switzerland

accepted on the recommendation of
Prof. Dr. Lothar Thiele, examiner
Prof. Dr. Hynek Hermansky, co-examiner
Dr. Berthold Crysmann, co-examiner
Dr. Beat Pfister, co-examiner

2009
Acknowledgements

I would like to thank Prof. Dr. Lothar Thiele and Dr. Beat Pfister for supervising my research, and Prof. Dr. Hynek Hermansky and Dr. Berthold Crysmann for taking part in my examination committee. This work would not have been possible without the support of the Swiss National Science Foundation, which is gratefully acknowledged.

I am indebted to numerous people who have contributed to this work. First of all, I wish to thank René Beutler, with whom I shared the pleasure of advocating a rather unpopular idea. His enthusiasm was a great source of motivation and inspiration. I am also grateful to Thomas Ewender for a somewhat shorter but nonetheless fruitful collaboration, and to Dr. Jean-Luc Gauvain for providing me with output of the LIMSI broadcast news transcription system. Finally, I would like to thank Dr. Berthold Crysmann for his detailed and helpful comments on an earlier version of this thesis, and Sarah Hoffmann for the thorough proofreading\(^1\).

\(^1\)I have inserted a bonus typo as a reward to the attentive reader.
## Contents

List of Abbreviations ................................................. 9

Abstract ............................................................ 11

Kurzfassung .......................................................... 13

1 Introduction ......................................................... 15
   1.1 Problem Statement ........................................... 15
   1.2 Scientific Contributions ...................................... 17
   1.3 Structure of the Thesis ....................................... 17

2 Describing Natural Language ..................................... 19
   2.1 Linguistic Concepts and Terminology ....................... 19
   2.2 Formal Grammars ............................................. 21
   2.3 Statistical Parsers ........................................... 24
   2.4 Discussion ................................................... 26

3 Structure in Language Modeling ................................ 29
   3.1 Language Modeling and ASR .................................. 29
   3.2 Generative Language Models .................................. 31
   3.3 Discriminative Language Models ............................. 33
   3.4 Statistical Parsers as Language Models ..................... 34
   3.5 Formal Grammars as Language Models ....................... 36
   3.6 Discussion ................................................... 43

4 Integrating Linguistic Constraints ............................. 47
   4.1 Architecture .................................................. 47
   4.2 Discriminative Reranking ..................................... 53
   4.3 Disambiguation ............................................... 58
      4.3.1 Robust Parsing .......................................... 58
      4.3.2 Implementation Details .................................. 62
      4.3.3 Features ................................................ 63
   4.4 ASR Hypotheses Reranking ................................... 71
   4.5 Segmentation ................................................ 75
   4.6 Discussion ................................................... 76

5 Linguistic Subsystem ............................................... 79
   5.1 The Grammar Formalism ...................................... 79
      5.1.1 A Very Short Introduction to HPSG ..................... 79
      5.1.2 Characteristics of the Grammar Formalism ............. 84
   5.2 The Grammar System .......................................... 88
      5.2.1 Architecture ............................................ 88
      5.2.2 The Description Language ............................... 90
   5.3 The Parser ................................................... 92
      5.3.1 Parsing Algorithm ....................................... 93
      5.3.2 Optimizations ............................................ 95
      5.3.3 License Rules .......................................... 96
   5.4 The Grammar ................................................ 104
      5.4.1 Adjuncts and Complements ............................... 104
      5.4.2 Verbal Projections ...................................... 105
      5.4.3 Noun Phrases ............................................ 114
      5.4.4 Coordination ............................................ 116
      5.4.5 Extrapolation ............................................ 117
      5.4.6 Subgrammars ............................................ 119
   5.5 The Part-of-Speech Tagger ................................... 120
   5.6 Discussion ................................................... 122

6 Development of Linguistic Resources .......................... 125
   6.1 Introduction ................................................ 125
   6.2 Acquisition of Open-Class Words ............................ 126
### List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR</td>
<td>automatic speech recognition</td>
</tr>
<tr>
<td>AVM</td>
<td>attribute-value matrix</td>
</tr>
<tr>
<td>CFG</td>
<td>context-free grammar</td>
</tr>
<tr>
<td>HMM</td>
<td>hidden Markov model</td>
</tr>
<tr>
<td>HPSG</td>
<td>head-driven phrase structure grammar</td>
</tr>
<tr>
<td>MAP</td>
<td>maximum-a-posterior</td>
</tr>
<tr>
<td>NLP</td>
<td>natural language processing</td>
</tr>
<tr>
<td>NP</td>
<td>noun phrase</td>
</tr>
<tr>
<td>PCFG</td>
<td>probabilistic context-free grammar</td>
</tr>
<tr>
<td>PP</td>
<td>prepositional phrase</td>
</tr>
<tr>
<td>SER</td>
<td>sentence error rate</td>
</tr>
<tr>
<td>VP</td>
<td>verb phrase</td>
</tr>
<tr>
<td>WER</td>
<td>word error rate</td>
</tr>
</tbody>
</table>
Abstract

Large-vocabulary continuous speech recognition relies on prior knowledge about a natural language to complement the acoustic models. N-grams, the most widely used language models, largely disregard the structure of natural language. In the last decade, progress in the field of statistical parsing has led to the development of more powerful statistical language models that also consider syntactic structure. These models have shown that information about the structure of natural language can significantly improve the accuracy of automatic speech recognition.

Unlike statistical parsers, formal grammars are designed to discriminate between grammatical and ungrammatical sentences. Formal grammars have been successfully applied to narrow-domain natural language understanding tasks, but not to broad-domain speech recognition. In fact, it appears that grammar-based approaches to speech recognition do not easily scale up to broad domains. Some of the main difficulties that have to be faced are lack of precision, lack of coverage and a reduced benefit from pure grammaticality information.

The aim of this thesis is to demonstrate that hard linguistic constraints represented in a formal grammar can improve automatic speech recognition on a broad domain. To this end, a novel approach to integrating formal grammars into large-vocabulary continuous speech recognition is proposed. This approach is based on a discriminative reranking scheme that considers syntactic features of the N best speech recognition hypotheses. The syntactic features are extracted by means of a precise formal grammar (a Head-driven Phrase Structure Grammar) with a stochastic disambiguation component.

The feasibility of our approach is verified experimentally. For a German broadcast news transcription task, we report a statistically significant reduction of the word error rate by 1.3% absolute (9.7% relative) compared to a competitive baseline system, namely the LIMSI German broadcast news transcription system. To our knowledge, this is the first significant improvement on a broad-domain speech recognition task due to a formal grammar. Different properties of the proposed approach are investigated in a series of additional experiments.
Kurzfassung


Im Gegensatz zu den statistischen Ansätzen sind formale Grammatiken darauf ausgelegt, grammatische von ungrammatischen Sätzen zu unterscheiden. Formale Grammatiken wurden erfolgreich für das automatische Sprachverstehen in relativ eingeschränkten Anwendungsbereichen eingesetzt, nicht aber in der automatischen Spracherkennung für allgemeinere Domänen. Tatsächlich scheint es, dass sich bestehende Ansätze zur grammatischen Spracherkennung nicht leicht auf allgemeinere Domänen übertragen lassen. Dabei sind die Hauptprobleme die Präzision der Grammatik, die Abdeckung der auftretenden grammatischen Phänomene und die verrinerte Aussagekraft grammatischer Korrektheit.


Es wird gezeigt, dass dieser Ansatz sowohl realisierbar als auch nutzbringend ist. Bei der automatischen Transkription deutscher Nachrichtensendungen konnte die Wortfehlerrate um 1.3% absolut (9.7% relativ) verringert werden. Diese Verbesserung bezieht sich auf ein konkurrenzfähiges Spracherkennungssystem, das LIMSI German broadcast news transcription system. Für die Spracherkennung in einer allgemeinen Domäne ist dies meines Wissens die erste signifikante Verbesserung, die mit Hilfe einer formalen Grammatik erzielt wurde. In einer Reihe von zusätzlichen Experimenten werden verschiedene Eigenschaften des vorgeschlagenen Ansatzes untersucht.
Chapter 1

Introduction

1.1 Problem Statement

Acoustic models in automatic speech recognition have difficulty in distinguishing between alternative transcriptions that are phonetically similar. In our German experimental data, for example, the correct transcription “das ist besonders im Schwarzwald so” is judged to be acoustically less likely than the completely ungrammatical transcription “dass es besonders im Schwarzwald so”. The role of a language model is to complement the acoustic model with prior information about the word sequences that occur in a particular language.

The most widely used class of language models, the so-called n-grams, are based on the assumption that the probability of a word only depends on a small number of preceding words. These models can easily be trained on large amounts of unannotated text, and they perform extremely well despite of their simplicity.

However, n-grams fail to capture many dependencies that are present in natural language. In our experimental data, the n-gram language model prefers the incorrect transcription “einer, der sich für die Umwelt einsetzt”. This transcription is ungrammatical because the number of the relative pronoun der (singular) does not agree with the number of the finite verb einsetzen (plural). In order to capture this instance of subject-verb agreement, an n-gram would have to consider at least five words preceding the verb. Moreover, the training corpus would have to contain the correct word sequence “der sich für die Umwelt einsetzt”, which is quite unlikely. We do not claim that n-grams and variants such as skip n-grams or class-based n-grams cannot possibly deal with this particular example. But it is easy to find many similar examples for any language model that does not properly capture the dependencies of natural language.

In the last decade, progress in the field of statistical parsing has led to the development of more powerful language models that also incorporate syntactic structure. These language models have been shown to increase recognition accuracy for broad-domain speech recognition tasks.

Formal grammars, which are designed to discriminate between grammatical and ungrammatical sentences, have been successfully applied to narrow-domain natural language understanding tasks, but not to broad-domain speech recognition. This may partly be due to the difficulty of scaling such approaches to broad domains. Precise broad-domain grammars are notorious for their lack of coverage and at the same time fail to exclude many incorrect but nevertheless grammatical transcriptions. This greatly diminishes the value of grammaticality information. In addition, the development of grammars and lexica for broad domains is a difficult and laborious task. All these effects can be controlled much better for restricted domains.

This situation is somewhat unsatisfying as formal grammars do have properties that are interesting for language modeling. They appear to be the most adequate means to formalize hard linguistic constraints. If they are complemented with statistical models for disambiguation, these models typically require relatively few syntactically annotated training data compared to statistical parsers. And finally, formal grammars encode linguistic constraints that are – at least to some degree – orthogonal to word-based statistical models, including n-grams and statistical parsing models.

In this thesis, we investigate whether broad-domain large-vocabulary speech recognition can benefit from formal grammars. In particular, we propose and evaluate an approach of integrating formal grammars into speech recognition. We do not aim for an efficient real-time system, but rather for a deeper understanding of how linguistic information can contribute to speech recognition.
1.2 Scientific Contributions

The following contributions resulted from the present thesis:

1. We have proposed a novel approach of integrating formal grammars into a statistical speech recognition framework. This approach is based on discriminative reranking and robust parsing.

2. We have used our approach to extend a competitive baseline speech recognition system. To our knowledge, we report the first statistically significant improvement on a broad-domain speech recognition task due to a formal grammar.

3. We have empirically investigated various properties of our approach, for example the impact of different linguistic information and the types of errors that are corrected by the grammar-based system.

4. Additional contributions are related to diverse problems that were encountered in the course of this work. They include parsing techniques and methods for the development of broad-coverage grammars and lexica.

1.3 Structure of the Thesis

The remainder of this thesis is structured as follows:

Chapter 2 introduces the relevant linguistic concepts and terminology and presents some of the formal tools that are used for modeling natural language.

Chapter 3 gives a short introduction to language modeling in automatic speech recognition and surveys previous approaches of using syntactic structure in language modeling. Finally, challenges for grammar-based approaches are discussed.

Chapter 4 presents an outline of the proposed approach and describes some of its components.

Chapter 5 is dedicated to the linguistic components of the proposed approach. The linguistic components were developed in the course of this thesis and comprise a German grammar, a parser and a part-of-speech tagger.

Chapter 6 is concerned with the development of large-coverage grammars and lexica. We present our approach and discuss some of the problems which have been encountered.

Chapter 7 describes our experiments and evaluates different aspects of the proposed approach.

Chapter 8 concludes the thesis with a final discussion.
Chapter 2
Describing Natural Language

The aim of this chapter is to introduce some basic concepts and terminology that are relevant for the description of natural language. In particular, fundamental notions from the areas of linguistics, formal grammars and statistical parsing are explained. The final section will discuss the benefits and weaknesses of formal grammars in the context of natural language processing.

2.1 Linguistic Concepts and Terminology

This section introduces the specific linguistic terminology that is used throughout this thesis. At the same time, a few general linguistic concepts and natural language phenomena are presented. These phenomena may give an idea of the linguistic constraints and the dependencies that should be considered in formal and statistical accounts of natural language.

A phrase is a syntactic unit which is postulated in a given syntactic theory or grammar. Common examples of phrases in English are noun phrases like “the heady seduction of vanilla and cedarwood, underlining the presence of today’s woman in all her touching complexity”. A phrase may be composed of embedded phrases such as the prepositional phrase “of vanilla and cedarwood” in the above example. Such embedded phrases are also called constituents in order to emphasize the compositional aspect.

The head of a phrase is the constituent which can be seen as primarily determining the syntactic properties of the phrase. For example, the head of a noun phrase is the noun and the head of a sentence is the finite verb, i.e. the verb that indicates present or past tense in English. A phrase that is headed by a certain word (its lexical head) is also called a projection of that word.

Besides the head constituent, a phrase may also contain several complements and adjuncts of the head. Complements are constituents that are selected (or required) by the head. For example, subjects and objects are complements of verbs, and prepositions have a noun phrase as their complement. The set of complements of a head is also termed the subcategorization or valence of that head. A phrase is called saturated if it contains all complements of its head. For example, the nominal projection “der kleine Baum” is saturated, whereas “kleine Baum” is not.

Adjuncts or modifiers select the head constituent and contribute to the semantics of the phrase. They are optional from a syntactic point of view, and they can occur arbitrarily often in a phrase. Prototypical examples are adjectives and adverbs. The term adjunct is used to emphasize the syntactic aspect, whereas “modifier” is a semantic notion.

Words and phrases can be classified with respect to the roles they can take on within a sentence. The individual classes are termed syntactic categories. Parts-of-speech such as noun, adjective or verb are possible, though rather coarse syntactic categories. More informative syntactic categories may also represent specific agreement features (e.g. case, number and gender) or even valence information.

Extraction and extraposition are two general phenomena that occur in natural languages. In extraposition, a phrase is detached from its head and moved to the end of the sentence. An example of extraposition is “a man was arrested who was carrying a violin case”, where the relative clause is detached from its head, the noun man. Extraction refers to a process by which a constituent is moved out of a phrase, possibly crossing several clause boundaries. For example, in “what do you want me to talk about?” the interrogative pronoun what can be thought of as having been moved out of the phrase “about what” to the beginning of the sentence.
A precise model of natural language has to take into account phenomena such as subcategorization, extraction or extraposition and the involved dependencies. Otherwise, the model may not be able to reflect the fact that the following sentences are perceived as being incorrect: “he slept around the garden”, “where do you want me to talk about”, “two men were arrested who was carrying a violin case”.

2.2 Formal Grammars

After this brief introduction to linguistic notions and phenomena, we turn to the formal means for capturing them. We will first introduce some basic concepts of the theory of formal grammars. Then, it will be shown how formal grammars are used to model natural language.

The Chomsky Hierarchy

A formal grammar is a set of rules that describes whether a string of symbols is syntactically valid or not. Thus, a grammar implicitly defines a set of syntactically valid (or grammatical) strings. This set is also called the language of the grammar. It is common to say that a grammar accepts strings from its language and rejects any other string. Strings that are rejected by the grammar are also termed out-of-grammar.

[Cho56] distinguished four types of formal grammars that differ in their expressive power: the formal grammars of one type describe a proper subset of those languages that can be expressed by the next lower type. A Chomsky grammar employs a set of rewrite rules (production rules) that operate on strings of terminal symbols and non-terminal symbols. The set $V_T$ of terminal symbols is the alphabet of the language. The non-terminal symbols defined by the set $V_N$ can be regarded as auxiliary symbols. A sequence of terminal symbols is accepted by a grammar if and only if the production rules allow to rewrite a dedicated start symbol as this target sequence. Each type of formal grammar is subject to specific restrictions on the form of the production rules:

Regular grammars (type 3):

\[ A \rightarrow aB, \quad A \rightarrow a \quad \text{or} \quad A \rightarrow \epsilon \]

or

\[ A \rightarrow Ba, \quad A \rightarrow a \quad \text{or} \quad A \rightarrow \epsilon \]

for $a \in V_T$ and $A, B \in V_N$.

Context-free grammars (type 2):

\[ A \rightarrow \psi \quad \text{for} \quad A \in V_N \quad \text{and} \quad \psi \in (V_N \cup V_T)^* \]

Context-sensitive grammars (type 1):

\[ \varphi_1 A \varphi_2 \rightarrow \varphi_1 \psi \varphi_2 \quad \text{for} \quad A \in V_N \quad \text{and} \quad \varphi_1, \varphi_2, \psi \in (V_N \cup V_T)^* \]

Unrestricted grammars (type 0):

\[ \varphi \rightarrow \psi \quad \text{for} \quad \varphi \in (V_N \cup V_T)^+ \quad \text{and} \quad \psi \in (V_N \cup V_T)^* \]

In the above notation, a production rule of the form $A \rightarrow \epsilon$ rewrites the non-terminal symbol $A$ as an empty sequence of symbols.

Formal Grammars and Natural Language

If formal grammars as defined above are used to describe natural language, the terminal symbols typically represent word forms and the language of the grammar is the set of grammatical sentences. The notion of grammaticality is somewhat ambiguous in this context: a sentence can be grammatical with respect to a formal grammar or with respect to the natural language that is being modeled.
considered to be sufficiently powerful for modeling natural language, even though a few languages (including a dialect of Swiss German) are provably not context-free. These languages exhibit so-called cross-serial dependencies and thus belong to the class of mildly context-sensitive languages.

Nevertheless, the CFG formalism is generally considered inappropriate for writing precise large-coverage grammars, i.e. grammars that cover a large fragment of a natural language and at the same time reliably exclude ungrammatical sentences. As is argued in [SW99, Chapter 2], this is less a matter of formal expressiveness than of descriptive adequacy. Precise, large-coverage grammars require rich sets of highly specific syntactic categories. These categories typically include agreement features, valence and additional information for describing movement phenomena such as extraction and extraposition. If such syntactic categories are represented by atomic non-terminal symbols, the grammar cannot directly model the general, relatively independent phenomena introduced in the previous section. As a consequence, the grammar will be massively redundant.

There are a number of grammar formalisms that offer formal means to represent rich syntactic categories and to capture general linguistic phenomena. Examples of such formalisms are Dependency Grammar [Tes59], Definite Clause Grammar (DCG) [PW80], Lexical Functional Grammar (LFG) [Bre82] and Head-driven Phrase Structure Grammar (HPSG) [PS87]. A short introduction to Head-driven Phrase Structure Grammar will be given in Section 5.1.1. DCG, LFG and HPSG are unrestricted grammars (Chomsky type 0) and thus can, in theory, simulate arbitrary Turing machines. However, this does not mean that this computational power needs to be fully exploited for modeling natural language. For example, the formalism of Generalized Phrase Structure Grammar [GKPS82] attempts to provide adequate descriptive means without going beyond the class of context-free languages.

So far, formal grammars were only considered as a means to decide whether a sentence is grammatical or not. However, many applications of natural language processing are more interested in the actual syntactic structure of a sentence. A parser is a program which determines the grammaticality of a sentence and at the same time computes all possible derivations of that sentence. A derivation is a sequence of rule applications and corresponds to a specific syntactic structure. Derivations can also be represented as parse trees. Due to the ambiguity of natural language, there are in general many possible derivations for a grammatical sentence. Ambiguities are typically resolved by means of a statistical model that defines a probability distribution over the infinite set of derivations. In probabilistic context-free grammars (PCFGs), each rule is assigned a probability. The probability of a derivation is computed by iterating over all rule applications and multiplying the respective rule probabilities. A more general approach to statistical disambiguation will be described in Section 4.3.

2.3 Statistical Parsers

The primary aim of a statistical parser is to provide a linguistically plausible parse tree for any given word sequence. Unlike formal grammars, statistical parsers are not designed to discriminate between grammatical and ungrammatical word sequences. Rather, they are tailored to the processing of naturally occurring language, including all the irregularities, idiosyncrasies and errors that are frequently observed in real-world text.

The need for robustness warrants a stochastic approach to parsing. Statistical parsers model the relation between word sequences \( W \) and parse trees \( T \) as a probability distribution \( P(T|W) \) or \( P(T,W) \). The actual parser implements a search strategy for finding the best parse tree \( \hat{T} \) for a given word sequence:

\[
\hat{T} = \underset{T}{\text{arg max}} \ P(T|W) = \underset{T}{\text{arg max}} \ P(T,W) \tag{2.1}
\]

The parameters of the probability distribution are estimated on large syntactically annotated corpora, so-called treebanks. One of the most widely used corpora for English is the Penn treebank [MMS93]. For some statistical parsers, additional expectation-maximization-like training was performed on unannotated data. For example, [Cha97] reports modest gains of accuracy due to unsupervised training.

Although standard PCFGs may seem to provide a reasonable approach for modeling \( P(T,W) \), they have turned out to be rather poor models for the distribution of sentences. One reason for this is the fact that the context-freeness assumption is too strong. For example, [Kle03] note that in English, a subject noun phrase is 8.7 times more likely to expand into a pronoun than an object noun phrase. Further, a finite set of grammar rules is inevitably incomplete, which may prevent some of the observed
word sequences from being parsed. This is also the case for so-called
treebank grammars, whose rules are directly read off a treebank (see e.g.
[Cha97]).

In order to increase the amount of context information, most statis-
tical parsers define a set of parser operations such that each sequence of
operations \( \langle o_1, o_2, ..., o_n \rangle \) uniquely produces a parse tree \( T \) and a word
sequence \( W \). Using the chain rule, \( P(T, W) \) can then be rewritten as follows:

\[
P(T, W) = P(\langle o_1, o_2, ..., o_n \rangle) = \prod_{i=1}^{n} P(o_i|\langle o_1, o_2, ..., o_{i-1} \rangle)
\]  

(2.2)

The conditioning information \( \langle o_1, o_2, ..., o_{i-1} \rangle \) is called the history at the
decision point \( i \). This history-based modeling approach was introduced
in [BJL+93] and adopted in virtually all state-of-the-art parsers [Col99,
Rat99, Cha00, Roa01c, CJ05]. Note that a PCFG is in fact a history-
based model which assumes that the probability of a parser operation
(in this case a rule expansion) only depends on the non-terminal to be
expanded. In a top-down derivation, this non-terminal is part of the
derivation history.

The history-based approach allows to consider much more powerful
conditioning information. One source of information which is exploited
by most statistical parsers are so-called head words. The head word of a
phrase is the word which can be regarded as characterizing the phrase.
Following an example by [Cha97], profits is the head word of the phrase
“corporate profits”, and rose is the head word of the sentence “corporate
profits rose”. If, for instance, a distribution of rule expansions is condi-
tioned on the involved head words, the model might judge \( S \to NP \ VP \)
to be more likely if \( NP \) is headed by \( profits \) and \( VP \) is headed by \( rose \). However, there is some evidence that the value of head word information
has been overestimated in the past. For example, [Bik04] has shown that
dilexical information (i.e. the conditioning on two head words) adds very
little to the performance of the Collins parser [Col99]. [Kle03] have devel-
oped a statistical parser that does not use head words (except for function
words) and yet is surprisingly accurate.

In order to guarantee complete coverage of all possible input word
sequences, many statistical parsers employ so-called Markov grammars
[Col99, Roa01c, Cha00]. A Markov grammar defines an infinite number
of context-free rules which can be thought of as being generated by a ran-
don process. In particular, the non-terminal sequence on the right-hand
side of the rule is generated by a Markov process. For example, [Col99]
models the probability of a rule expansion \( A \to L_N \ldots L_1 H R_1 \ldots R_M \)
as follows:

\[
P(L_N \ldots L_1 H R_1 \ldots R_M | A) = P(H|A) \times \prod_{i=1}^{N} P(L_i |_i \ldots L_{i-K+1}, H, A) \times \prod_{i=1}^{M} P(R_i | R_{i-1} \ldots R_{i-K+1}, H, A)
\]

The non-terminal \( H \) represents the head-child of the phrase \( A \). The non-
terminals to the left and to the right of \( H \) are generated by two \( K \)th
order Markov processes that are conditioned on the non-terminal to be
expanded and the characterizing child \( H \). Of course, all distributions are
in practice further conditioned on information from the derivation history.

The accuracy of statistical parsers can be further improved by letting
the parser produce the \( N \) most probable parse trees for a given word
sequence and then using discriminative reranking to pick the best of these
parses [CK05, CJ05]. Discriminative reranking allows to introduce very
diverse information that goes beyond the conditioning information of the
statistical parser. This technique will be discussed in Section 4.2.

2.4 Discussion

Probabilistic approaches to natural language processing have clearly be-
come predominant in the last two decades. This seems to contradict the
tradition of formal grammar, where a sentence typically is considered to
be either grammatical or ungrammatical, but nothing in between:

\[
\text{But it must be recognized that the notion 'probability of a sen-
tence' is an entirely useless one, under any known interpreta-
tion of this term. [Cho68, p. 57]}
\]

In fact, there is evidence that the boundary between hard constraints on
grammaticality and mere preferences is blurred. [Man03] notes that peo-
ple “continually stretch the rules of grammar to meet new communicative
needs”. In particular, he argues that there is no sharp distinction between
complements and modifiers and that speakers can easily deviate from a
verb’s standardly assumed valence properties. For instance, although the
verb *to quiver* is generally considered to be intransitive, [AL95] observed
instances of transitive use such as “(...) it runs along in a zigzag path,
quivering its wings (...).” The justified criticism of the categorical view of
natural language has even led to the rather controversial claim that the
notion of grammaticality should be abandoned altogether [Sam07].

On the other hand, [Kel00] performed a series of experiments which
suggest that human sentence processing does involve hard constraints in
addition to soft constraints. In these experiments, subjects were asked to
direct the acceptability of sentences, some of which violated certain lin-
guistic constraints. It was observed that the violation of some constraints
(which intuitively would be classified as hard constraints) severely hurt
the acceptability, whereas the violation of the other constraints resulted
in a mild unacceptability.

If hard constraints are relevant for describing natural language, it ap-
pears to be reasonable to formalize such constraints and exploit this in-
formation in natural language processing. Yet the utility of formal gram-
mars for practical applications is often doubted, mostly because of their
assumed lack of robustness and processing cost. However, this might be a
misconception. For example, [KRK+04] reported that their parsing sys-
tem, an LFG parser with a stochastic disambiguation component and a
simple robustness scheme, accurately identified semantically relevant de-
pendencies in newspaper text. In fact, their system was more accurate
than a state-of-the-art statistical parser (the Collins parser) with only a
modest increase in parsing time.

Following [Meu07], we think that many – though by no means all –
aspects of syntax are best described by means of hard constraints. We
further believe that such constraints (as encoded in a formal grammar)
can be exploited in real-world tasks if appropriately complemented with
soft constraints and a way of dealing with violated hard constraints. In
particular, we will show that hard constraints are beneficial in speech
recognition, where constraint violation can help to distinguish between
correct and incorrect hypotheses.
Chapter 3
Structure in Language Modeling

This chapter briefly outlines the standard approach to automatic speech recognition and describes how language models can be applied within this framework. Next, existing approaches for integrating syntactic information in language modeling are surveyed. These approaches fall into two major classes: those based on statistical parsing and those based on formal grammars. The chapter concludes with a discussion of the presented approaches.

3.1 Language Modeling and ASR

In the most general sense, a language model encodes prior information about a natural language. Language models are used in various applications of natural language processing. As we are mostly concerned with automatic speech recognition, this section will give a brief introduction to the standard approach to speech recognition. In particular, we will point out how language models are used within this approach.

A speech recognition system basically transforms an acoustic observation $X$ (a speech signal) into a word sequence $W$ (the transcription). In order to measure the performance of a speech recognizer, some sort of evaluation metric is required. The most commonly used metrics are the word error rate (WER) and the sentence error rate (SER). The word error rate is computed as the Levenshtein distance [Lev66] between the automatic transcriptions and the reference transcriptions, normalized by the total length of the reference transcriptions. The sentence error rate is the number of automatic transcriptions that completely match their corresponding reference transcription, divided by the total number of transcriptions.

The standard approach to automatic speech recognition is based on the maximum-a-posterior (MAP) rule [BJM83]. Given an acoustic observation $X$, the MAP rule states that the optimal transcription $\hat{W}$ is the word sequence $W$ maximizing $P(W|X)$, the posterior probability of $W$ given $X$:

$$\hat{W} = \arg\max_W P(W|X).$$  \hspace{1cm} (3.1)

It can be shown that if $\hat{W}$ is chosen according to the MAP rule, the sentence error rate is minimized. Using Bayes’ theorem, the posterior probability can be rewritten as

$$\hat{W} = \arg\max_W \frac{P(X|W) \cdot P(W)}{P(X)}. \hspace{1cm} (3.2)$$

The denominator $P(X)$ can be omitted as it does not depend on $W$:

$$\hat{W} = \arg\max_W P(X|W) \cdot P(W). \hspace{1cm} (3.3)$$

$P(X|W)$ represents the acoustic model and $P(W)$ the language model. Although $P(W)$ is in fact a generative language model (see next section), it is simply termed “the language model” in the context of MAP-based speech recognition. The role of the language model is to compensate for inaccuracies of the acoustic models and to resolve ambiguities that are inherent to spoken language.

Although the decision rule from Equation (3.3) is theoretically optimal, it is commonly extended with a language model weight $\lambda$ and a word insertion penalty $ip$. These modifications have turned out to improve the recognition performance in practice by compensating for inadequacies of the acoustic model and the language model.

$$\hat{W} = \arg\max_W P(X|W) \cdot P(W)^\lambda \cdot |W|^{ip}. \hspace{1cm} (3.4)$$

In order to maximize the right-hand side expression, the speech recognizer has to perform a search in a vast space of hypotheses. This search
Generative Language Models

In generative language modeling, the prior information about language is represented as a probability distribution \( P(W) \) over the (infinite) set of word sequences \( W = \langle w_1, w_2, \ldots, w_K \rangle \). Actual models of \( P(W) \) are often derived from the chain rule decomposition:

\[
P(W) = \prod_{k=1}^{K} P(w_k | w_1 \ldots w_{k-1})
\]

3.2 Generative Language Models

In generative language modeling, the prior information about language is represented as a probability distribution \( P(W) \) over the (infinite) set of word sequences \( W = \langle w_1, w_2, \ldots, w_K \rangle \). Actual models of \( P(W) \) are often derived from the chain rule decomposition:

\[
P(W) = \prod_{k=1}^{K} P(w_k | w_1 \ldots w_{k-1})
\]

3.2 Generative Language Models

In generative language modeling, the prior information about language is represented as a probability distribution \( P(W) \) over the (infinite) set of word sequences \( W = \langle w_1, w_2, \ldots, w_K \rangle \). Actual models of \( P(W) \) are often derived from the chain rule decomposition:

\[
P(W) = \prod_{k=1}^{K} P(w_k | w_1 \ldots w_{k-1})
\]
The \textit{n-gram} language model approximates \( P(W) \) by assuming that the probability of a word only depends on the \( n-1 \) preceding words:

\[
P_n(W) = \prod_{k=1}^{K} P(w_k|w_{k-n+1} \ldots w_{k-1}) \quad (3.6)
\]

N-grams constitute the predominant approach to language modeling in statistical speech recognition. The success of n-grams is due to several reasons. First of all, n-grams can directly be incorporated into Viterbi decoding. This allows to use language model information at a very early stage of the search. Further, n-gram distributions can be estimated from unannotated text, which is often available in large quantities. As n-gram probabilities are based on actual word forms rather than part-of-speech tags or more fine-grained syntactic categories, they also capture certain semantic aspects of word sequences.

On the other hand, it was demonstrated in Section 1.1 that syntactic dependencies are not limited to a window of \( n \) words. In addition, n-gram language models are subject to the problem of data sparsity: the training data cannot be expected to cover all sequences of \( n \) words that may occur in actual use. The sparsity problem can partly be attributed to the fact that n-grams do not reflect the hierarchical and combinatorial nature of language.

### 3.3 Discriminative Language Models

Following the general framework of [Roa09], a discriminative language model is trained to choose the best output \( y^* \), given a set of output candidates \( \text{GEN}(x) \) for the input observation \( x \). In the context of automatic speech recognition, \( x \) is an acoustic observation and \( \text{GEN}(x) \) typically denotes the \( N \) best speech recognition hypotheses.

The model parameters are in general not directly optimized to minimize the word error rate. However, the actual objective functions are closely correlated with word error rate. A common example of such an objective is to maximize the conditional likelihood \( \prod_i P(y_i^*|\text{GEN}(x_i)) \), where \( y_i^* \) denotes the candidate with the lowest word error rate for the \( i^{th} \) training example.

One of the strengths of discriminative language models is that they are based on arbitrary real-valued features that are extracted from each candidate \( y \in \text{GEN}(x) \). These features allow to integrate heterogeneous information such as acoustic likelihoods or discrete linguistic events. However, this property is not exclusive to discriminative language models. For example, the whole-sentence language model [RCZ01] is a generative language model that offers the same flexibility.

Section 4.2 will provide a more detailed discussion of a particular class of discriminative language models that are based on so-called conditional log-linear models.

### 3.4 Statistical Parsers as Language Models

Statistical parsers that model the joint distribution \( P(T, W) \) of the parse tree and the word sequence can straightforwardly be turned into a generative language model \( P(W) = \sum_T P(T, W) \). As statistical parsers typically employ a beam search, the summation takes place over the most probable parse trees only. Statistical parsers that process sentences from the left to the right [CJ00, Roa01b, WH03, vUC05] are particularly interesting because they can compute the probability of an arbitrary prefix of \( W \). This in turn allows to compute the probability of a word given the preceding words, \( P(w_k|w_1, w_2, \ldots, w_{k-1}) \), which can be used to guide the search in a word lattice.

The structured language model [CJ00] is based on a non-deterministic shift-reduce parser that operates from left to right. For each prefix string \( w_1, w_2, \ldots, w_{k-1} \), it generates a sequence of binary-branching partial parse trees in a bottom-up manner. The probability of the next word \( w_k \) is conditioned on the head words (and the respective part-of-speech tags) of the two rightmost trees. Thus, a structured language model is a kind of trigram that looks at the head words of consecutive phrases rather than at consecutive words. The structured language model was applied to speech recognition by means of an A* search in the word lattice. [CJ00] reported a reduction of the word error rate on three tasks: read Wall Street Journal text, Switchboard (spontaneous telephone speech) and broadcast news transcription. For the Wall Street Journal task, the word error rate was further reduced by [XCJ02]. They used a 50-best rescoring approach and an improved version of the structured language model. Interestingly, they outperformed the parser-based approach by [Roa01b] even though their parsing accuracy was much worse than that of the latter system.
3.4 Statistical Parsers as Language Models

[Roa01b] presented a probabilistic left-to-right parser with a context-free Markov grammar. As the parser employs a top-down parsing strategy, it can compute the actual PCFG probability of a prefix word sequence. This stands in contrast to the structured language model which models probability distributions by conditioning on parser operations. [Roa01b] achieved a reduction of the word error rate on the Wall Street Journal task by rescoring the 50-best hypotheses. [Roa01a] proposed a pruning technique that allowed to apply his parser to the efficient rescoring of word lattices. However, he observed only a marginal reduction of the word error rate in comparison to N-best rescoring.

[Cha01] used the Charniak parser as a language model. The parser first generated candidate trees which were subsequently re-evaluated by means of an elaborate statistical model. The model conditioned the probability of a rule expansion on the head word of the corresponding phrase. This prohibits left-to-right parsing, as a left-to-right parser will in general not yet have encountered the head word at the time it has to decide on the rule expansion. [HJ03] reported that this language model outperformed the models of [CJ00] and [Roa01b] on the Wall Street Journal task when N-best rescoring is employed. They also proposed a method of applying the Charniak parser to word lattices. In short, they used a parser for context-free Markov grammars that searched the word lattices for promising parse trees, guided by a sophisticated figure of merit. A small number of parse trees was passed to the Charniak parser, which assigned them a new probability. This lattice-based approach was found to be more efficient than N-best rescoring, however at the cost of a substantially decreased recognition performance.

[CRS05] applied the Collins parser to a speech recognition task. Rather than computing \( P(W) \), they extracted the most likely parse tree for each of the N best hypotheses. Subsequently, they extracted syntactic features from the parse trees and fed them into a discriminative language model which reranked the N best hypotheses. Their experiments on the Switchboard corpus showed modest reductions in word error rate. These reductions were mostly due to features based on part-of-speech tags, which did not involve the parse trees at all. [MKHO06] used the Charniak parser in a similar setup. They reported a significant reduction in word error rate on Switchboard data. The results are not comparable to those of [CRS05], as the latter used a baseline speech recognition system with a much higher first-best word error rate.

A completely different approach was taken by [WH02, WSH04], whose language model is essentially a class-based n-gram. Their approach is relevant to the present discussion because their classes, the so-called SuperARVs, precisely describe the syntactic properties of words. SuperARVs are derived from corpora that are annotated with Constraint Dependency Grammar (CDG) parses. They specify information such as agreement features, the word’s required dependants and the relative positions of these dependants. In fact, an assignment of SuperARVs to the words in a sentence almost determines the parse structure. Such an assignment is thus termed almost-parsing, following the terminology from the related field of supertagging [BJ99]. Due to its n-gram structure, the SuperARV language model can be applied for lattice rescoring and it can even be integrated into the decoding pass. It has been successfully applied to the Wall Street Journal task [WH02] and to the Switchboard data [WSH04].

[WH03] extended the almost-parsing approach of [WH02] to a complete statistical CDG parser by assigning dependency links between SuperARVs in a probabilistic way. As their generative model can compute the probabilities of prefix word sequences, it can be used for lattice rescoring. On the Wall Street Journal task, they achieved a reduction of the word error rate that is comparable to the results for the revised structured language model [XCJ02].

3.5 Formal Grammars as Language Models

Whereas it is conceptually straightforward to integrate a statistical parser into MAP-based speech recognition, it is much less evident how to exploit the constraints that are encoded in a formal grammar. We will next survey approaches that use formal grammars for language modeling in speech recognition. The discussion is structured into five main aspects: the speech recognition tasks, the different ways of integrating formal grammars into speech recognition, properties of the formal grammars, robustness techniques and the impact on the speech recognition performance.
Tasks

Virtually all work discussed in this section is targeted at speech understanding in rather small domains. Many of the earlier approaches were applied to the DARPA 1000-word resource management task [PFBP88]. Later tasks include the MIT Voyager system [ZGG91], the ATIS air travel information system [Pri90] and the OVIS public transport information service [NBKv97]. The only systems that were not developed for a spoken dialog application are Verbmobil [Wah00], a speech-to-speech translation system in the domain of appointment scheduling and travel planning, and the grammar-based speech recognizer by [BKP05a]. The task of the latter system was the transcription of dictation texts for pupils.

Integration into Speech Recognition

Some of the early approaches to grammar-based speech recognition fall outside the class of speech recognizers that was outlined in Section 3.1 in that they do not employ HMM state transition networks and Viterbi decoding. [EHLR80] and [KS89] proposed architectures where an acoustic module and a parser explore the search space in an alternating fashion. If there is strong acoustic evidence that a certain word or word sequence covers a certain time interval, the acoustic module (the so-called wordspotter) generates a corresponding hypothesis. The parser can extend a hypothesis by predicting words that can precede or follow the hypothesis according to the grammar. These predictions are subsequently verified and evaluated by the acoustic module. In this scheme, the grammar constrains the possible word sequences and helps to spot words that are notoriously difficult to detect in speech signals, e.g. short function words. [Nak89] proposed a similar approach in which the grammar can only predict function words.

In [KKS89], the search is dominated by the parser. The parser iteratively generates prefix word sequences that are consistent with the grammar. The time alignment and the acoustic match of a prefix word sequence are efficiently computed by means of dynamic programming. The parser performs a beam search in the space of prefix word sequences, retaining only those hypotheses with the highest acoustic match. [Ney91] computes the globally optimal word sequence, word boundaries and parse tree in a single dynamic programming pass. The grammar is assumed to be context free.

The approaches of [GSHP91] and [God92] are based on stack decoding rather than Viterbi decoding. Stack decoding is essentially an A* search for the most likely word sequence. At each iteration, the most likely partial hypothesis is retrieved from a priority queue and extended with possible successor words. Each of the resulting hypotheses is assigned a score and put into the priority queue. [GSHP91] and [God92] used left-to-right parsers to incrementally process the partial hypotheses, i.e. the respective prefix word sequences. [GSHP91] restricted the set of possible successor words to those accepted by the grammar, whereas [God92] estimated the probability of a prefix word sequence. The language model of [God92] can be regarded as a deterministic, unlexicalized version of the structured language model: the probability of the next word was conditioned on the top 2 stack elements of a deterministic shift-reduce parser.

The remaining approaches are extensions of the standard speech recognizer architecture sketched in Section 3.1. As was already pointed out, this architecture facilitates the integration of language models at three main levels: the decoder, the word lattice and the N best hypotheses. If the integration takes place on a lower level (i.e. in an earlier stage of the processing), the language model can support hypotheses that would not have been available on the next higher level due to pruning. On the other hand, the lower levels impose stronger restrictions on the structure and the processing efficiency of the language model.

One way to integrate grammar constraints into the decoding pass is to specify the topology of the speech recognizer’s state transition network. [MDB+97] converted an application-specific unification grammar into an equivalent regular grammar, which was possible because the grammar made very limited use of recursion. A regular grammar can be directly compiled into a state transition network.

Non-regular grammars can be integrated by dynamically extending the state transition network. One approach is to let a left-to-right parser predict the set of possible word transitions whenever the final state of a word is reached [MPM89]. A variant of the same idea is to let the parser prune partial hypotheses as soon as they undergo invalid word transitions [MM90b, KW91]. Both approaches need to memorize the parser states after the processing of each partial hypothesis. This can be achieved by associating a set of parser states with each network state [MPM89], or
by interpreting network states as positions in the parser chart [MM90b]. [JWS+95] computed the exact probability of the next word given a prefix word sequence and a PCFG. This probability was derived from PCFG probabilities of prefix word sequences, which in turn can be obtained by means of left-to-right parsing [Sto95]. The approximate word transition probabilities at a given state were computed from the N most likely prefix word sequences for that state.

A particularly elegant way of integrating CFGs into the decoder was proposed by [Dup93]. His approach seamlessly integrates parsing into dynamic network generation. For each non-terminal, the set of possible expansions is encoded as a state transition network. The non-terminals that may appear in these expansions are represented as non-terminal transitions. Whenever the decoder encounters a non-terminal transition, the transition is replaced by the network of the corresponding non-terminal. This method is only applicable for grammars that are not left-recursive. Fortunately, any CFG can be transformed into an equivalent CFG without left recursion.

A completely different approach is to approximate a grammar with a simpler language model that can easily be applied in the decoding pass. [ZGG+91] used a unification grammar to generate random text from which feasible word pairs were extracted. This word pair language model was essentially a bigram that rejects some of the hypotheses that will never lead to a grammatical word sequence. These local grammaticality constraints were complemented with N-best parsing for determining the best grammatical hypothesis. [SS94] proposed an algorithm for computing the precise n-gram probabilities from a PCFG. PCFG bigrams were applied to speech recognition by [SS94] and [JWS+95].

Exhaustive parsing of word lattices was adopted in [CS89], [NBKv97], [HWW+00] and [van01]. In order to facilitate exhaustive processing, word lattices are sometimes pruned [van01] or transformed into alternative representations such as word graphs [JH99] or minimized deterministic weighted finite-state automata [MR97]. Such representations are less redundant from the parser's point of view. [Kie05] proposed an island parsing algorithm for word graphs, however without reporting actual speech recognition experiments. The algorithm allows to parse word graphs by bidirectionally extending partial parse trees, starting at nodes with high certainty. [BKP05a, Beu07] used the N best hypotheses to guide non-exhaustive lattice parsing. They started with a trivial word graph that contained only the best hypothesis. This word graph was incrementally extended with the nodes and edges of the next best hypothesis. After each extension, a parser was run to process the new paths.

The lattice parsing scheme of [KKK+99] amounts to sequentially parsing the N best hypotheses. However, the parser chart reflects the topology of the word lattice. This information allows to reuse partial derivations for common substrings. Finally, [ZGG+91] and [MAD+95] employ genuine N-best parsing.

**Grammars**

Most of the discussed approaches are based on context-free grammars or formally equivalent variants that allow for non-terminals with atomic-valued features.

Some approaches made use of sophisticated unification grammars that model more involved phenomena such as non-local dependencies. Examples are the TINA grammar [Sen92] employed in the MIT Voyager system [ZGG+91, GSHP91] and the Gemini grammar which was applied to the ATIS task [MAD+95]. The Verbmobil grammar [MK00] and the grammar used in [BKP05a] are instances of the HPSG formalism. Both grammars cover a relatively large fragment of German. However, they fail to model certain grammatical constructions that would be important for parsing broad-domain text. [Müll04] notes that the Verbmobil grammar did not cover optionally coherent verbs and certain types of extraposition, among others. The grammar by [BKP05a] includes these phenomena but does not cover constructions such as appositions and sentential complements of nouns. The Dutch OVIS grammar [vBKN99] is a unification grammar in the spirit of HPSG. It was not intended as a wide-coverage grammar and cannot account for passives, relative clauses and certain types of verb clusters.

Some speech understanding systems use semantic constraints that restrict the grammar to accept only those sentences which are meaningful in the given domain. Semantic constraints can be integrated into a grammar by encoding semantic information in non-terminal symbols [KW91, JWS+95] or features [Sen92, MAD+95, MDB+97]. They can

---

1 As the random sampling does not guarantee that all feasible word pairs are found, some grammatical word sequences may be rejected as well.
Robustness

Most of the surveyed work simply chooses the best hypothesis (with respect to the acoustic model and possibly a statistical language model) that is accepted by the grammar. Such an approach is bound to fail in the following situations:

1. The correct hypothesis is accepted but there is another accepted hypothesis with a higher recognition score. If hypotheses are only compared with respect to grammaticality and recognition score, the incorrect hypothesis will be preferred.

2. None of the considered hypotheses are accepted by the grammar. In this case, it is not possible to improve on the top-ranked hypothesis and no semantic interpretation can be provided. If the grammar guides the search, no hypotheses will be produced at all.

3. The correct hypothesis is not among the considered hypotheses or it is not accepted by the grammar. The more hypotheses are considered, the more likely it is that some low-ranked hypothesis will be accepted instead, potentially causing a large number of word errors.

The first problem can be handled by taking the probability of a derivation into account. [JWS+95] and [RHB+06] employed probabilistic context free grammars, whereas [GSHP91] made use of the stochastic component of the TINA NLP system. All of these systems integrate the grammar in the decoding pass. Thus, the main objective of the probabilistic models is to guide the search of the speech recognizer.

There are several ways of dealing with the second problem. In [KW91] and [JWS+95], partial hypotheses are parsed incrementally as they grow from left to right. If at some point all partial hypotheses fail to parse, bigram probabilities are used to score partial hypotheses for the remainder of the utterance. [God92] predicts the next word from the current state of a deterministic shift-reduce parser. If a word transition produces an unparseable partial hypothesis, the parser state is reset. The new parser state is chosen such that the parser tries to parse a specific constituent, depending on the last word transition.

Impact on Recognition Performance

Some studies report a WER reduction compared to a bigram [KW91, God92], trigram [van01] or 4-gram language model [BKP05a]. However, the relevance of these results is unclear. The only statistically significant improvements relative to an acknowledged state-of-the-art baseline system were reported by [MAD+95]. They achieved a relative WER reduction of 14.6% on an ATIS task. The baseline system was the DECIPHER ATIS speech recognition system that performed best in the December 1993 ATIS SPREC evaluation. For the Verbmobil system, effects on the speech recognition performance were not reported.
3.6 Discussion

In this chapter, we have surveyed two areas of research that are concerned with introducing syntactic structure into language modeling. The two areas differ in the formal tools they use to capture syntactic constraints: statistical parsers or formal grammars, respectively.

Statistical parsers are natural candidates for language modeling as they are robust with respect to ungrammatical utterances and unknown words, and they can be straightforwardly turned into generative language models. They have been successfully applied to broad-domain speech recognition tasks such as the transcription of read newspaper text, broadcast news or spontaneous telephone conversations, yielding statistically significant improvements.

It is much less obvious how formal grammars can be integrated into a probabilistic speech recognition framework. Formal grammars have almost exclusively been applied to tasks that involve natural language understanding in rather restricted domains. The single exception is the work by [BKP05a, Beu07], which is targeted at speech recognition rather than understanding. Their approach was based on an earlier version of the grammar which will be presented in this thesis. They reported a significant WER reduction on an artificial speech recognition task, namely the transcription of dictation texts for pupils. This task was relatively easy due to the simple language, the small recognizer vocabulary (which completely covered the test utterances) and the good acoustic conditions.

There are several reasons why formal grammars are attractive for small-domain speech understanding tasks. First, if parsing is necessary for extracting the semantic interpretation of an utterance, there is no significant overhead in applying the same tools and resources to speech recognition. Further, small domains may allow for very restrictive grammars that constrain both the syntax and the semantics of the accepted utterances. Restrictive grammars are more effective in identifying the correct recognizer hypotheses. In fact, [RHB^*06] created natural language understanding systems by compiling a general grammar into more restricted, domain-specific grammars. For small domains, restrictive grammars can still have a reasonably good coverage. In applications of speech-based human-computer interaction, the user can even be expected to adapt to the grammar with increasing experience. Finally, there is often not enough domain-specific data for the training of statistical language models. This makes grammar-based language models an appealing alternative.

In broad-domain large-vocabulary speech recognition, it is much more difficult to exploit the strengths of formal grammars. The processing of broad-domain text requires both a general grammar and a general lexicon. Apart from increasing ambiguity, this has the effect that many incorrect hypotheses are accepted by the grammar. Example (3.7) shows some incorrect but grammatical hypotheses from our test data. For each hypothesis, a word-to-word translation and a full translation are provided. The correct hypothesis was “arafat räumte fehler seiner regierung ein” (arafat admitted mistakes of his government).

(3.7) (a) arafat räumte fehler seiner regierung allen
      arafat cleared mistakes his government everyone
      ’arafat cleared mistakes of his government for everyone’
(b) arafat räumte fehler seiner regierung allein
      arafat cleared mistakes his government by himself
      ’arafat cleared mistakes of his government by himself’
(c) arafat wollte fehler seiner regierung einen
      arafat wanted mistakes his government to unify
      ’arafat wanted to unify mistakes of his government’
(d) arafat warnte fehler seiner regierung allein
      arafat warned mistakes his government by himself
      ’arafat warned mistakes of his government by himself’
(e) arafat und der fehler seiner regierung allein
      arafat and the mistake his government only
      ’only arafat and the mistake of his government’

Some of these sentences are clearly marked, but they are perfectly possible in appropriate contexts and with a semantically plausible choice of words. Sentence (3.7) (c) is accepted because of the general lexicon: lexica for narrow domains would most probably list the word einen as a determiner, but not as a verb. Sentence (3.7) (e) contains a post-nominal focus particle (allein), which is unlikely to be covered by a narrow-domain grammar.

It is clear from this example that simply choosing the acoustically best grammatical hypothesis will not work for broad-domain tasks. Rather,
there should be a component that captures syntactic preferences and allows to compare different grammatical hypotheses. The problem of accepting incorrect hypotheses becomes even more difficult if the grammar is not precise, i.e. if the grammar is not designed to reliably reject incorrect sentences. Thus, a precise grammar appears to be preferable for the given task, even though a strong model for syntactic preferences may partly compensate for overgeneration of the grammar.

A problem which is converse to the productivity of general grammars is their lack of coverage: formal grammars that aim at precision in broad domains are inevitably incomplete. On the one end of the spectrum, there are general grammatical constructions that are notoriously difficult to formalize, for example ellipses or asymmetric coordinations such as “I am neither an authority on this subject nor trying to portray myself as one” [SGWW85]. On the other end, there is a wealth of idiosyncratic phenomena such as determiner omission in temporal expressions like “nächste Woche” (next week).

But even if standard language were completely covered by the grammar, we would still have to deal with non-standard language use and ungrammatical utterances. In addition, the speech recognizer vocabulary cannot be expected to cover all words (in particular proper names) that occur in the utterances, and the sentence boundaries may be inaccurate if they are determined automatically. As a consequence, the correct hypothesis will often be out-of-grammar or it will not be present in the reduced search space at all. This further aggravates the problem of incorrect but grammatical hypotheses, and it necessitates the use of a robustness component.

In summary, we have argued that a broad-domain speech recognition task requires a general but precise grammar (and hence a formalism that facilitates the development of such a grammar), a strong model for syntactic preferences and a robust way of integrating these components into speech recognition. None of the surveyed approaches to speech recognition with formal grammars meet these requirements. As a matter of fact, none of these approaches were reported to have yielded significant improvements on a broad-domain speech recognition task.

Considering the interesting properties of formal grammars (the adequate formalization of hard constraints, the smaller amounts of training data and the orthogonality to word-based statistical approaches; see Section 2.4), it seems worthwhile to investigate the integration of formal
Chapter 4

Integrating Linguistic Constraints

This chapter will first provide a brief overview of the proposed approach and then describe some of the relevant components in more detail. Finally, the approach will be discussed and compared to related work. The concepts presented in this chapter are largely (although not completely) independent of the actual grammar and parser. The linguistic components will be described in Chapter 5.

4.1 Architecture

In this section, we will outline our approach to integrating a formal grammar into speech recognition. As was argued in the previous chapter, such an approach should consider syntactic preferences and it should be robust with respect to out-of-grammar hypotheses. We will show that our approach provides solutions for both of these problems.

The approach is divided into two stages. In the first stage, a speech signal is processed by a baseline speech recognizer and the resulting word lattice is split into sub-lattices that represent sentence-like units. This procedure is illustrated in Figure 4.1. The second stage, sketched in Figure 4.2, is applied to each sub-lattice. First, the N best hypotheses are extracted from the given sub-lattice. Each hypothesis is parsed and the resulting syntactic information is used to choose the most promising hypothesis.

The benefit of the linguistic post-processing can easily be assessed by comparing the word error rate of the baseline system to that of the complete system. Note that the baseline speech recognizer already employs a statistical language model. In the present approach, this language model is complemented with a discriminative language model that considers syntactic information. In the following, some aspects of the two processing stages will be described in greater detail.

Segmentation

The segmentation into sentence-like units is necessary because the original segments of the baseline speech recognizer can be very long. In our experiments, they cover up to 60 seconds of speech or 23 sentence units. Segments of that length are prohibitive for parsing. They also result in an impoverished search space: if a long segment is divided into a sequence of M shorter segments, a reranking approach can choose M times among N hypotheses. Thus, the hypothesis space for the original segment is increased from N to NM hypotheses.
4.1 Architecture

Segment boundaries should not break sentences apart. For a segment that does not match the sentence boundaries, the best available hypothesis is likely to be ungrammatical. This implies that syntactic information can only be extracted from the major constituents of the hypothesis. Thus, the segment boundaries are chosen to match automatically determined sentence boundaries. If a segment is too long, it is split at a point that is expected to impair the syntactic analysis the least. The details of the segmentation algorithm and the automatic sentence boundary detection are given in Section 4.5.

N-Best Hypothesis Reranking

After segmentation, each sub-lattice is processed as shown in Figure 4.2. First, the N best hypotheses are extracted from the sub-lattice. Each hypothesis consists of a word sequence and a baseline speech recognizer score, which is in turn computed from the respective acoustic likelihood and the score of the statistical language model. The word sequence of the best hypothesis is exactly the one that would have been produced by the baseline speech recognizer. Table 4.1 shows the five best hypotheses for the utterance “als Bush im Auto den Reichstag verliess, konnte er sicher sein” (when Bush left the Reichstag by car, he could be sure). The fifth best hypothesis is correct.

In the next step, a parser is used to derive all possible partial parse trees for each hypothesis. Partial parse trees are parse trees that cover arbitrary parts of the hypothesis word sequence. A partial parse tree does not necessarily need to represent a sentence, but it is reasonable to specify a restricted set of admissible syntactic categories. The partial parse trees of a hypothesis are represented as a packed parse forest [Tom91].

A packed parse forest preserves the syntactic ambiguities of a hypothesis. The role of the disambiguation component is to resolve all of these ambiguities and to provide a unique complete parse tree for each hypothesis. In particular, the disambiguation component should produce a complete parse tree even if there is no partial parse tree that covers the whole hypothesis word sequence. In order to achieve this, we introduce the notion of an artificial parse tree.

Table 4.1: The five best hypotheses for the utterance “als Bush im Auto den Reichstag verliess, konnte er sicher sein”. Each hypothesis is assigned a score $s_{asr}$ by the baseline speech recognition system. This score includes the acoustic log-likelihood and logarithmic n-gram probabilities. The correct hypothesis is ranked fifth.

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Word Sequence</th>
<th>$s_{asr}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>als burschen autor den reichstag verliess konnte er sicher sein</td>
<td>$s_{asr} = -614265.7$</td>
</tr>
<tr>
<td>2.</td>
<td>als burschen auto den reichstag verliess konnte er sicher sein</td>
<td>$s_{asr} = -614275.3$</td>
</tr>
<tr>
<td>3.</td>
<td>als puschen autor den reichstag verliess konnte er sicher sein</td>
<td>$s_{asr} = -614290.1$</td>
</tr>
<tr>
<td>4.</td>
<td>als wuschen autor den reichstag verliess konnte er sicher sein</td>
<td>$s_{asr} = -614290.5$</td>
</tr>
<tr>
<td>5.</td>
<td>als bush im auto den reichstag verliess konnte er sicher sein</td>
<td>$s_{asr} = -614291.3$</td>
</tr>
</tbody>
</table>

Figure 4.2: Second stage: The N best hypotheses are extracted and the most promising hypothesis is chosen based on syntactic information and the scores assigned by the baseline speech recognizer.
Figure 4.3 shows two examples of artificial parse trees. An artificial parse tree is created by attaching a number of partial parse trees to a common root node. If there is a partial parse tree that spans the entire hypothesis, the disambiguation component can (but need not) use this partial parse tree as the single child of the root node. This case is illustrated in the lower part of Figure 4.3, which shows an artificial parse tree for the fifth-best solution from Table 4.1. The upper part shows an artificial parse tree for the first-best hypothesis, for which no complete parse tree exists. In such cases, the disambiguation component determines the most plausible sequence of partial parse trees and combines these trees to an artificial parse tree. Disambiguation is based on a statistical model that assigns a score to an artificial parse tree. Given a hypothesis word sequence, the disambiguation component selects the artificial parse tree with the highest score. The disambiguation component and the statistical model will be explained in greater detail in Section 4.3.

The scores produced by the statistical model cannot only be used to compare alternative artificial parse trees for a given hypothesis. They also allow to compare the most plausible artificial parse trees of different hypotheses. Table 4.2 shows the five best hypotheses of our example with both the baseline speech recognizer score and the disambiguation score. In this example, the correct hypothesis on the fifth rank achieves the highest disambiguation score. The corresponding artificial parse tree is the one shown in the lower part of Figure 4.3.

In order to decide which hypothesis to choose, different pieces of evidence have to be taken into account. Besides the acoustic score, the n-gram language model score and the disambiguation score, we also consider prosodic cues and different features extracted from the most plausible artificial parse trees. This evidence is combined by means of a log-linear model, resulting in a final score for each hypothesis, the recognition score. Discriminative reranking chooses the hypothesis with the highest recognition score. The log-linear model and the individual features are detailed in Section 4.4.

To conclude this brief overview, Table 4.3 shows the final recognition score for the five best hypotheses of our example. The highest recognition score is achieved by the fifth hypothesis, which (of course) happens to be the correct one.

Figure 4.3: Two artificial parse trees resulting from the disambiguation process. The first tree consists of five partial parse trees attached to a common root node. The second tree consists of a single complete parse tree.
4.2 Discriminative Reranking

Disambiguation and choosing the most plausible speech recognition hypothesis are both based on discriminative reranking with conditional log-linear models. This section will give a short introduction to conditional log-linear models and how they are applied to discriminative reranking.

Conditional Log-linear Models

Conditional log-linear models were first introduced to the natural language processing community by [BDD96]. They represent a family of conditional probability distributions of the following form:

\[ P_\theta(y|x) = \frac{1}{Z_\theta(x)} e^{\sum_j \theta_j f_j(x,y)} \]  

(4.1)

Table 4.2: The five best hypotheses from Table 4.1 with their respective disambiguation scores \( s_{\text{dis}} \). The correct hypothesis on rank 5 receives the highest disambiguation score.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Hypothesis</th>
<th>( s_{\text{asr}} )</th>
<th>( s_{\text{dis}} )</th>
<th>( s_{\text{rec}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>als burschen autor den reichstag verliess konnte er sicher sein</td>
<td>-614265.7</td>
<td>-0.09</td>
<td>-41418.00</td>
</tr>
<tr>
<td>2</td>
<td>als burschen auto den reichstag verliess konnte er sicher sein</td>
<td>-614275.3</td>
<td>-0.11</td>
<td>-41420.06</td>
</tr>
<tr>
<td>3</td>
<td>als puschen autor den reichstag verliess konnte er sicher sein</td>
<td>-614290.1</td>
<td>-1.72</td>
<td>-41420.46</td>
</tr>
<tr>
<td>4</td>
<td>als wuschen autor den reichstag verliess konnte er sicher sein</td>
<td>-614290.5</td>
<td>-2.51</td>
<td>-41420.06</td>
</tr>
<tr>
<td>5</td>
<td>als bush im auto den reichstag verliess konnte er sicher sein</td>
<td>-614291.3</td>
<td>+4.16</td>
<td>-41416.90</td>
</tr>
</tbody>
</table>

Table 4.3: The five best hypotheses from Table 4.1 with the final recognition scores \( s_{\text{rec}} \). The hypothesis with the highest recognition score (the fifth-best hypothesis) is chosen as the final transcription. The recognition score is related to the logarithmic probability of a hypothesis according to a conditional log-linear model. This model integrates the disambiguation score \( s_{\text{dis}} \), the score of the baseline recognizer \( s_{\text{asr}} \) and a large number of additional features. The recognition score is discussed in Section 4.4.
In the above equations, \( y \in \mathcal{Y} \) is a class label and \( x \in \mathcal{X} \) is an observation to be classified. In the example of a simple part-of-speech tagger, \( \mathcal{Y} \) would be the set of part-of-speech tags and \( x \) would represent the word to be tagged. The feature functions \( f_j : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R} \) extract specific pieces of evidence from an observation. Even though feature functions can produce arbitrary real numbers, they typically count or indicate certain events that occur in the observation. For example, \( f_{\text{ends with "ity"}}(x, y) \) might be 1 if and only if the word to be tagged ends with “ity” and the part-of-speech tag \( y \) denotes a singular noun. The feature weights \( \theta_j \) quantify the strength of the evidence. If \( \theta_j \) is a large positive value, the feature function \( f_j(x, y) \) provides strong evidence in favor of class \( y \) in case \( f_j(x, y) > 0 \). Similarly, negative weights indicate negative evidence. \( Z_\theta(x) \) is a normalizing term which ensures that \( \sum_y P_\theta(y|x) = 1 \).

The model parameters \( \theta = (\theta_1, ..., \theta_n) \) are chosen to minimize the negative log-likelihood of the set of training samples \((x^{(s)}, y^{(s)})\):

\[
L(\theta) = -\log \prod_s P_\theta(y^{(s)}|x^{(s)})
\]

(4.3)

It can be shown that the loss function \( L(\theta) \) is convex and thus can be globally minimized by means of gradient descent methods. A log-linear model with appropriately chosen parameters has a number of interesting properties. First, the expectations of the feature functions \( f_j(x, y) \) with respect to \( P_\theta \) are identical to the expectations observed in the training data. Second, \( P_\theta \) is the only distribution that meets these expectation constraints and at the same time maximizes the entropy of the distribution [BDD96]. This means that the distribution is consistent with the facts present in the data but otherwise is as uniform as possible. In other words, the distribution does not make any assumptions beyond what is supported by the training data. Because of this second property, log-linear models are also termed maximum entropy models. A third property of log-linear models is that they do not make any independence assumptions: the feature functions can be arbitrarily correlated. This is particularly interesting for natural language processing, as it allows to specify features solely on linguistic grounds without being limited by dependency considerations.

### Regularization

Even though the maximum entropy principle does have positive effects on the model’s generalization properties, conditional log-linear models are subject to overfitting. For instance, if a very specific feature happens to be non-zero for the correct class in one training example \((x^{(s)}, y^{(s)})\) but zero otherwise, the respective feature weight will approach infinity. Thus, such features are misinterpreted as perfect indicators for a given class.

A simple way of dealing with this problem is to remove constraints that lack support in the data, i.e. to remove features that are non-zero less than \( K \) times [Rat96]. A technique which has a similar effect is feature selection, where a set of “most informative” features is determined with a greedy algorithm [BDD96]. The standard approach to prevent overfitting is regularization by means of prior distributions that penalize large feature weights. [CR99] introduced zero-mean Gaussian priors which penalize large feature weights and appear as quadratic terms in the negative log likelihood:

\[
L(\theta) = -\log \prod_s P_\theta(y^{(s)}|x^{(s)}) + \sum_j \frac{\theta_j^2}{2\sigma_j^2}
\]

(4.4)

An alternative approach is to use Laplacian priors [Goo04], for which the regularization term contains \(|\theta_j|\) rather than \(\theta_j^2\). Several studies give theoretical and empirical evidence that Laplacian priors (or \(l_1\) regularizers) are superior to Gaussian priors [Goo04, RV04, Ng04, DPS04]. However, as the derivative of \(|\theta_j|\) is discontinuous, standard gradient-based techniques cannot be applied directly to the minimization of the loss function.

In order to train our models, we apply the MaxEnt reranker presented in [CJ05]. As this software only supports Gaussian priors, we do not make use of Laplacian priors. We further follow [CJ05] in using a single regularization constant \(c\) instead of the \(1/2\sigma_j^2\), which results in the regularization term \(c\sum_j \theta_j^2\). The constant \(c\) has to be optimized on held-out data.

### Reranking

Reranking applies to situations where an observation and several candidate interpretations are given. The goal of reranking is to find the most likely interpretation for a given observation. [JGC+99] proposed a reranking approach based on conditional log-linear models. In the notation of
In contrast to equations 4.1, 4.2 and 4.3, the observation \( x \) is now represented by the set \( \mathcal{Y} \) of all hypotheses that are consistent with \( x \). The features are only extracted from the hypotheses because the hypotheses can be thought of as containing all relevant information about the observation. As with the original version of conditional log-linear models, efficient methods for minimizing the loss function exist [Mal02].

The training data consists of pairs \((y^{(s)}, \mathcal{Y}^{(s)})\) where \( y^{(s)} \in \mathcal{Y}^{(s)} \) is the correct (or best) hypothesis from the set \( \mathcal{Y}^{(s)} \). If the training data annotations are not specific enough, several hypotheses may conform to the reference annotation. [RKK+02] proposed a variant of the above approach which allows to specify several best hypotheses for each training example. However, we do not make use of this extension.

Given a set of appropriately trained feature weights \( \theta_j \), the most likely hypothesis \( y^{*} \) can be obtained as follows:

\[
y^{*} = \arg\max_y P_\theta(y|\mathcal{Y}) = \arg\max_y \sum_j \theta_j f_j(y)
\]

Thus, a hypothesis \( y \) maximizes the conditional probability \( P_\theta(y|\mathcal{Y}) \) if and only if it maximizes the term \( \sum_j \theta_j f_j(y) \). This term will subsequently be called the score of hypothesis \( y \).

An alternative to reranking with conditional log-linear models is a discriminative reranking technique based on the averaged perceptron algorithm [FS99]. This technique is similar to the one just discussed in that the best hypothesis is the one maximizing a weighted sum of feature values. The main difference is the algorithm for estimating the feature weights. The averaged perceptron is reported to perform worse than conditional log-linear models [RSCJ04], but it has the advantage of producing less non-zero feature weights.

### 4.3 Disambiguation

The goal of our disambiguation component is to provide a unique parse tree for arbitrary word sequences. In order to disambiguate between alternative parse trees, we employ discriminative reranking based on conditional log-linear models as introduced in the previous section. This can be regarded as common practice when dealing with unification-based grammars [RKK+02, Mv04].

However, the disambiguation component should also produce a reasonable parse tree even if no complete parse tree exists or if none of the complete parse trees matches the correct syntactic structure. As an attempt to deal with this problem, we have developed a novel robust parsing technique that is tightly integrated into stochastic disambiguation. This approach will be presented in the next section. Finally, we will discuss the set of features used in our log-linear model and a few implementation issues.

#### 4.3.1 Robust Parsing

Robust parsing has to deal with situations where the parser fails to derive the correct syntactic structure for a given word sequence. One situation of this kind occurs if there is no complete parse tree at all. In this case, the disambiguation component should identify as many correct partial parse trees as possible. A similar situation arises if there is one or more complete parse tree, but none of them matches the correct syntactic structure. Here, the disambiguation component should prefer plausible partial parse trees to a complete but very unlikely parse tree.

As mentioned in Section 4.1, our approach to robust parsing is based on the notion of an artificial parse tree. An artificial parse tree is created by attaching partial parse trees to a common root node and thus is equivalent to a sequence of partial parse trees. Two examples of artificial parse trees are shown in Figure 4.3 on page 52. Note that complete parse trees are considered to be special instances of a partial parse trees. As every word is a (degenerate) partial parse tree, there exists at least one artificial parse tree for any word sequence.

In our approach to robust parsing, the problem of identifying the most likely sequence of partial parse trees is reformulated as that of finding the most probable artificial parse tree. A conditional log-linear model is
used to estimate the probability of an artificial parse tree given the set of all artificial parse trees that are consistent with the word sequence. A major difficulty of this approach is that in general, the number of artificial parse trees is exceedingly high: the rather short utterance "guten abend meine Damen und Herren" gives rise to roughly 85,000 artificial parse trees, whereas the numbers for some of the longer utterances exceed the capacity of a 64-bit integer. Thus, it is infeasible to enumerate all artificial parse trees. This complicates the search for the most likely artificial parse tree, but also the training of the conditional log-linear model. An efficient search algorithm and a feasible training procedure will be presented in the following sections.

Search Algorithm

Let the artificial parse tree \( y_a \) be composed of the partial parse trees \( p_1, p_2, \ldots, p_n \). It was pointed out in Section 4.2 that maximizing the conditional probability of an interpretation \( y \) is equivalent to maximizing \( \sum_j \theta_j f_j(y) \), which is denoted as \( \text{score}(y) \). The proposed search algorithm is based on the observation that \( \text{score}(y_a) = \sum_i \text{score}(p_i) \) if the features \( f_j(y) \) count linguistic events that are completely contained within the individual partial parse trees.

Under these assumptions, the most likely artificial parse tree can easily be determined by means of dynamic programming. For each subsequence \( w_a \ldots w_b \) of the given word sequence \( w_1 \ldots w_N \), the maximum score of all partial parse trees spanning \( w_a \ldots w_b \) is precomputed and stored as \( L(a, b) \). This is done by means of an ambiguity unpacking algorithm, as will be explained in Section 4.3.2. The recursive formula for computing the maximum score of all artificial parse trees is as follows:

\[
M(l, m) = \begin{cases} 
0 & \text{if } l = 0 \\
\max_{0 \leq k < l} (M(k, m) + L(k + 1, l)) & \text{if } 1 \leq l \leq N \\
-\infty & \text{otherwise}
\end{cases}
\]

The score of the most likely artificial parse tree is \( M(N) \). In order to recover the actual parse tree, all decisions taken in the computation of \( L \) and \( M \) have to be stored.

To make the model take into account the number of the partial parse trees, additional feature functions have to be specified. A feature function which counts the number of partial parse trees can easily be incorporated, as the event of a partial parse tree occurring is trivially contained within the partial parse trees. The respective feature weight is added to the score of each partial parse tree, which has the effect of penalizing artificial parse trees with a large number of partial parse trees. In the same way, it is possible to specify feature functions that count the number of partial parse trees with a certain syntactic category. Of particular interest are adverbials and nominative noun phrases. These phrases frequently occur in the kind of elliptical speech that is common for broadcast news reports.

To some extent, it is also possible to handle events that are not contained within the partial parse trees. For example, we use feature functions that are 1 if the number of partial parse trees lies within a certain interval, and 0 otherwise. An important feature of this family indicates the event that there are two or more partial parse trees. To deal with such features, additional state information (in this case \( m \), the number of partial parse trees) has to be included in the dynamic programming algorithm:

\[
M(l, m) = \begin{cases} 
0 & \text{if } l = 0 \text{ and } m = 0 \\
\max_{0 \leq k < l} (M(k, m - 1) + L(k + 1, l)) & \text{if } 1 \leq l \leq N \\
-\infty & \text{otherwise}
\end{cases}
\]

The score of the optimal artificial parse tree is \( \max_m M(N, m) + \delta(m) \), where \( \delta(m) \) is the contribution of those features that only depend on the number of partial parse trees. In a similar way, it would be possible to consider the relative order of the partial parse trees, or rather the order of the respective syntactic categories. For example, bigram feature functions could count pairs of adjacent syntactic categories. Such a feature would require to redefine \( L(a, b) \) as \( L(a, b, c) \), the score of the best partial parse tree that spans \( w_a \ldots w_b \) and has a certain syntactic category \( c \). We did not attempt to include this kind of feature, as the amount of training data was considered to be too small.

Training

It has already been pointed out that the large number of artificial parse trees complicates the training of a discriminative log-linear model. The same problem can arise for models that disambiguate between complete parse trees: if the grammar is not restrictive enough or the sentences
are too long, the set of candidate parse trees may become too large for training. [RKK+02] dealt with this problem by restricting the training to sentences with a manageable number of complete parse trees. [Osborn00] and [Mv04] considered all training sentences, but they created a candidate set by subsampling the full set of complete parse trees. [MT02] proposed a method for maximizing the likelihood of the training data without explicitly having to enumerate the complete parse trees. Their approach is based on efficiently computing the expectations of the feature functions from a packed parse forest.

The approach of [MT02] can in principle be adapted to our problem of training a disambiguation model for artificial parse trees. However, we followed [Osborn00] and [Mv04] in adopting a subsampling approach. This enabled us to use an available tool for the efficient training of discriminative reranking models, namely the reranker software by [CJ05]. Uniform random sampling of artificial parse trees would lead to impoverished candidate sets that are unlikely to contain many (if any) complete parse trees. We therefore resort to non-uniform sampling. A training example is constructed from 10 randomly selected artificial parse trees for every possible number of partial parse trees. The artificial parse tree that best matches the reference syntax tree is always included in the training example.

### Related Work and Discussion

A common approach to robust parsing is to choose a sequence of partial parse trees according to some heuristics. These heuristics typically minimize the number of partial parse trees and use some additional criteria in case of equality [van01, RKK+02, KRK+04, v+06]. Such fewest-chunks heuristics have the disadvantage that they prefer highly improbable parse trees if this results in a smaller number of partial parse trees. [Pv03] used part-of-speech tagger information to remove unlikely lexical items from the parser chart prior to parsing. This filter prevented the derivation of some unlikely partial parse trees, which in turn improved the accuracy of the fewest-chunks heuristics. However, unlikely partial parse trees need not necessarily involve incorrect part-of-speech tags, and categorical filtering may produce errors from which the parser cannot recover. [MAD+95] selected a sequence of partial parse trees according to the trigram probability of the respective sequence of syntactic categories. However, they did not consider probabilities of the individual partial parse trees.

[ZKF07] modeled sequences of partial parse trees by means of two distinct probability distributions. The first distribution models the partitioning of a given word sequence into segments, where each segment covers a single partial parse tree. The second distribution models the actual sequence of partial parse trees, given the partitioning of the word sequence. This latter distribution is based on a conditional log-linear model that estimates the probability of a partial parse tree. A drawback of this approach is that the model for the partitioning of word sequences involves a rather crude approximation. In addition, the authors did not provide an exact and efficient search algorithm for finding the optimal sequence of partial parse trees.

In the proposed approach, the preferences for selecting partial parse trees are integrated into the statistical disambiguation model. The model may prefer a non-minimal number of partial parse trees if this leads to a more likely artificial parse tree. As will be discussed in the next section, part-of-speech tagger information is also included in this model. However, this information is used as one cue among many others rather than as a categorical constraint.

In our experiments, the proposed approach improved only marginally upon the fewest chunks method (see Section 7.7.2). This may partly be due to the relatively small amount of training data or a suboptimal training method. This issue was not investigated any further.

### 4.3.2 Implementation Details

Almost all features used in our approach have the following form:

$$f_j(y) = \sum_{n \in \text{nodes}(y)} e_j(n) \tag{4.9}$$

In the above equation, \(y\) denotes a (partial or artificial) parse tree and \(\text{nodes}(y)\) is the set of nodes of the parse tree \(y\). The indicator function \(e_j(n)\) is 1 if a certain linguistic event is associated with node \(n\) and 0 otherwise. For example, if the feature \(f_j(y)\) counts the number of pronouns in parse tree, \(e_j(n)\) is 1 if and only if node \(n\) represents a pronoun.

The set of partial parse trees derived by the parser is usually represented as a packed parse forest [Tom91]. A packed parse forest describes an ambiguous structure by means of disjunctive nodes which represent
alternative instantiations of rule arguments. Enumerating all unambiguous parse trees that are implicitly represented in a packed parse forest is typically infeasible. Fortunately, there exist algorithms for ambiguity unpacking that extract the most likely parse trees from a packed parse forest without enumerating all parse trees. Some of these algorithms impose certain restrictions upon the indicator functions $e_j$.

The selective unpacking algorithm by [CO05] incrementally extracts parse trees in decreasing order of probability with respect to a log-linear model. The selective unpacking algorithm restricts the indicator functions $e_j(n)$ such that only nodes within a certain locality of $n$ can be examined for detecting linguistic events. More precisely, only subtrees of $n$ with a fixed height can be accessed. Larger subtree heights allow for greater flexibility in feature design but increase the computational complexity and the memory requirements. [ZOC07] described how the algorithm can be extended to support certain types of non-local features.

[Mv04] proposed a beam search algorithm which does not restrict the indicator functions, i.e. $e_j(n)$ can access the complete subtree of node $n$. However, the disadvantage of this algorithm is that it is not guaranteed to find the most likely parse tree.

We chose to use the selective unpacking algorithm in our experiments. The algorithm was slightly modified in order to reduce its memory requirements at the cost of obtaining only the most likely parse tree. The height of the local subtrees was chosen to be 3, i.e. four levels of nodes can be accessed.

### 4.3.3 Features

This section describes the features that are employed for disambiguation. Note that these features are specific to the language and (to a lesser extent) to the grammar used in our experiments. Many features are organized in so-called feature templates. An example is the feature template $np[D,H,M]$ shown in Table 4.4. A feature template represents a set of features, where each feature from the set corresponds to a particular instantiation of the template arguments. Thus, the template $np[D,H,M]$ produces, among others, feature which counts noun phrases with a determiner, a nominal head and without any modifiers.

| $np[D,H,M]$ | regular noun phrase |
| $D$ | determiner |
| no determiner |
| $H$ | head is a noun |
| head is a nominalized adjective |
| $M$ | no modifiers |
| one premodifier |
| two or more premodifiers |
| one postmodifier |
| one postmodifier and at least one additional modifier |

| $nn$ | regular noun phrase |
| $ne$ | proper name |
| pronoun | pronoun |
| pronoun-postmod | pronoun with postmodifier |
| pronoun-relative | relative pronoun |

Table 4.4: Basic noun phrase features.

Table 4.4 describes the basic noun phrase features which are designed such that they can be identified from a subtree of height 2. This makes it possible to use basic noun phrases as parts of more complex features.

Table 4.5 shows the set of generic rule features. The features of the first template count how often a certain grammar rule was applied in the derivation of the parse tree. The other features count the number of rule applications in different contexts such as specific instantiations of the head or non-head rule arguments or the length of the resulting constituent. The length-related features are not intended to precisely model the distribution of constituent lengths. Rather, they allow to penalize certain
4.3 Disambiguation

unlikely events such as long prenominal genitives or very short sentences. The cumulative binning approach used for modeling constituent lengths is considered to be robust in the face of mismatch between training and test data \cite{LSS+06, p. 1533}.

A related set of features describes sequences of rule applications. They apply to situations where some constituent is instantiated as the head argument of some rule, and the resulting constituent is again instantiated as the head argument of some rule. The features are defined by the following templates:

\[
\text{head-chain-2}[R_1, R_2] \rightarrow \text{projecting a head by subsequent application of rules } R_1 \text{ and } R_2
\]

\[
R_1 \quad \text{any grammar rule}
\]

\[
R_2 \quad \text{any grammar rule}
\]

\[
\text{head-chain-3}[R_1, R_2, R_3] \rightarrow \text{projecting a head by subsequent application of rules } R_1, R_2 \text{ and } R_3
\]

\[
R_1 \quad \text{any grammar rule}
\]

\[
R_2 \quad \text{any grammar rule}
\]

\[
R_3 \quad \text{any grammar rule}
\]

It can be observed that the disjuncts of coordinations tend to span the same number of words, which is particularly pronounced for coordinations of noun phrases and adjective phrases. This observation is captured by the feature template shown below. A similar feature template was proposed by \cite{CJ05}.

\[
\text{coord-parallelism}[H, D] \rightarrow \text{a coordination has two disjuncts with type } H \text{ and lengths } l_1 \text{ and } l_2, \text{ and } |l_1 - l_2| \leq D.
\]

\[
H \quad \text{any}
\]

\[
\text{noun phrase}
\]

\[
\text{adjective phrase}
\]

\[
D \quad \{0, 1, 2, 3, 5, 7\}
\]

The features shown in Table 4.6 give the model a rough notion of constituent order. They allow the model to learn, for example, that subjects and modifiers often occupy the Vorfeld and that the subject tends to precede the accusative object in the Mittelfeld.

\begin{table}[h]
\centering
\begin{tabular}{|l|}
\hline
rule[R] \rightarrow rule \( R \) is applied  \\
\( R \) \quad \text{any grammar rule}  \\
rule-np-head[R,H] \rightarrow rule \( R \) is applied to head constituent \( H \)  \\
\( R \) \quad \text{any grammar rule}  \\
\( H \) \quad \text{any basic noun phrase type from Table 4.4}  \\
rule-binary[R,H,N] \rightarrow binary rule \( R \) is instantiated with head \( H \) and non-head \( N \)  \\
\( R \) \quad \text{any binary grammar rule}  \\
\( H \) \quad \text{verb}  \\
\text{attributive adjective}  \\
\text{predicative/adverbial adjective}  \\
\text{preposition}  \\
\text{common noun}  \\
\text{proper name}  \\
\( N \) \quad \text{any basic noun phrase type from Table 4.4}  \\
\text{prepositional phrase}  \\
\text{adverb}  \\
\text{particle}  \\
\text{relative clause}  \\
\text{subordinate clause}  \\
\text{“dass” clause}  \\
\text{interrogative clause}  \\
\text{infinitive clause}  \\
\text{main clause}  \\
\rule-min-len[R,L] \rightarrow rule \( R \) creates phrase with at least \( L \) words  \\
\( R \) \quad \text{any grammar rule}  \\
\( L \) \quad \{2, 3, 4, 6, 8, 10, 15, 20\}  \\
\rule-max-len[R,L] \rightarrow rule \( R \) creates phrase with at most \( L \) words  \\
\( R \) \quad \text{any grammar rule}  \\
\( L \) \quad \{2, 3, 4, 6, 8, 10, 15, 20\}  \\
\hline
\end{tabular}
\caption{Generic rule features.}
\end{table}
Incorrect syntactic analyses often involve splitting noun phrases and analyzing the parts as individual noun phrases. The resulting noun phrases are often underspecified with respect to case. For example “die kleinen tassen” (the little cups) can be split into the pronoun die (nominative or accusative), the nominalized adjective kleinen (dative or accusative) and the noun tassen (nominative, dative or accusative). A common error is to analyze such a noun phrase as a dative complement of a verb\(^1\). The features described in Table 4.7 allow the model to penalize parse trees in which potential splitting artifacts are analyzed as complements of verbs.

In many incorrect syntactic analyses, words are assigned the wrong parts-of-speech. Examples for such analyses were given in the previous paragraph, where e.g. a determiner was mistaken for a pronoun. In order to avoid incorrect part-of-speech assignments, we use the output of a part-of-speech tagger. The following feature template penalizes deviations from the tagger output:

\[
\text{pos-mismatch}[P] - \text{a terminal node of the parse tree has the part-of-speech } P, \text{ but the part-of-speech tagger does not agree.}
\]

\[
P \text{ any}
\]

\[
\text{any pronoun proper name common noun compound part verb}
\]

In order to make these features less sensitive to tagging errors, the part-of-speech tagger generates all part-of-speech assignments that are at least half as likely as the most likely assignment. Thus, there can be several possible part-of-speech tags for each word. Only if none of these tags match the corresponding node of the syntax tree, the above features indicate a mismatch. The part-of-speech tagger is discussed in Section 5.5.

As mentioned in Section 4.3.1, [Pv03] also employ a part-of-speech tagger to reinforce parser preferences. However, they use the tagging information to remove non-conforming lexicon entries from the parser chart prior to parsing. Thus, the parser cannot recover from errors made by the

\(^1\)The dativus commodi and the dativus incommodi are treated as complements in the present grammar.
4.3 Disambiguation

**compcase**\([H,C]\) – a constituent with head \(H\) combines with a noun phrase complement required to have case \(C\).

- **H**
  - any
  - verb
  - adjective
- **C**
  - nominative
  - genitive
  - dative
  - accusative

**compcase-u**\([H,C,N]\) – a constituent with head \(H\) combines with a noun phrase complement required to have case \(C\). The case of the complement is underspecified to \(N\) possible cases.

- **H**
  - any
  - verb
  - adjective
- **C**
  - nominative
  - genitive
  - dative
  - accusative
- **N**
  - \{2, 3, 4\}

**compcase-t**\([G,T]\) – a constituent combines with a noun phrase complement of type \(T\) required to have case \(C\).

- **C**
  - nominative
  - genitive
  - dative
  - accusative
- **T**
  - any basic noun phrase type from Table 4.4

**Table 4.7:** Complement case features.

tagger. In our approach, the tagging information is integrated into the disambiguation model and can be overruled by other evidence.

Table 4.8 shows the set of features that are concerned with the number and types of partial parse trees. Note that some of syntactic categories listed for template **partial-parse-tree** are incomplete, namely the noun phrase with a missing determiner and the main clause with missing Vorfeld and missing subject. These syntactic categories are used to recover as much syntactic structure as possible from out-of-grammar word sequences. They are particularly useful when mandatory determiners are omitted and when the fronted subject and the finite verb are separated by a parenthesis.

| **partial-parse-tree**\([T]\) – partial parse tree of type \(T\) |
|---|---|
| **T** | any |
|   | trivial partial parse tree (single word) |
|   | adverbial |
|   | noun phrase |
|   | noun phrase (nominative) |
|   | noun phrase (missing determiner) |
|   | main clause |
|   | main clause (missing Vorfeld) |
|   | main clause (missing Vorfeld and missing subject) |
|   | infinitive clause |
|   | relative or interrogative clause |
|   | subordinate clause |

| **partial-parse-trees**\([N]\) – artificial parse tree with at least \(N\) partial parse trees |
|---|---|
| **N** | \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} |

**Table 4.8:** Partial parse tree features.
4.4 ASR Hypotheses Reranking

The goal of ASR hypotheses reranking is to choose the most promising speech recognition hypothesis for a given utterance. Each hypothesis is characterized by information from the baseline speech recognizer and by syntactic information extracted in the parsing and disambiguation phases. The information provided by the baseline system is the acoustic log-likelihood, the weighted n-gram language model score (including a word insertion penalty) and the overall speech recognition score which is simply the sum of the former two. The linguistic post-processing components contribute the most likely artificial parse tree and the corresponding disambiguation score.

In order to take a decision, we use a conditional log-linear model that estimates the conditional probability of each hypothesis. In fact, it is sufficient to compute the score \( \sum_j \theta_j f_j(y) \) of a candidate hypothesis \( y \), as was shown in Section 4.2. The log-linear model makes use of the following features \( f_j(y) \):

- the acoustic log-likelihood
- the n-gram language model score
- the overall score provided by the baseline speech recognizer
- the disambiguation score of the most likely artificial parse tree
- all disambiguation features defined in Section 4.3.3
- partial parse tree counts for non-first-best hypotheses
- a set of prosodic features

Some of the above features will next be discussed in greater detail. First, it should be noted that it is not clear what kind of information the disambiguation score represents in the present context. The score is not directly related to the conditional probability of the most likely artificial parse tree given the word sequence: the logarithm of this probability is

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pronoun-def-art-form</td>
<td>pronoun whose form matches that of a definite article</td>
</tr>
<tr>
<td>pronoun-indef-art-form</td>
<td>pronoun whose form matches that of an indefinite article</td>
</tr>
<tr>
<td>pronoun-relative[C]</td>
<td>relative pronoun with case ( C )</td>
</tr>
<tr>
<td></td>
<td>( C { \text{nominative, genitive, dative, accusative} } )</td>
</tr>
<tr>
<td>pronoun-def-art[C]</td>
<td>pronoun with case ( C ) whose form matches that of a definite article</td>
</tr>
<tr>
<td></td>
<td>( C { \text{nominative, genitive, dative, accusative} } )</td>
</tr>
<tr>
<td>pronoun-position</td>
<td>pronoun immediately follows a sentence-initial finite verb</td>
</tr>
<tr>
<td>adj-part[T]</td>
<td>adjectival participle of type ( T )</td>
</tr>
<tr>
<td></td>
<td>( T { \text{attributive, predicative/adverbial, any of the above} } )</td>
</tr>
<tr>
<td>adj-part-nom</td>
<td>nominalized adjectival participle</td>
</tr>
<tr>
<td>adj-part-aux-final</td>
<td>the adjectival participle precedes a finite sein/haben auxiliary verb in the right sentence bracket</td>
</tr>
<tr>
<td>adj-part-aux-initial</td>
<td>the adjectival participle precedes the empty right sentence bracket and the left sentence bracket consists of a finite sein/haben auxiliary verb</td>
</tr>
</tbody>
</table>

Table 4.9: Miscellaneous features.
equal to the disambiguation score minus a normalization offset that depends on the word sequence and thus is different for the individual speech recognition hypotheses. The disambiguation score could be interpreted as the logarithm of the conditional probability of the parse tree, given the parse trees of all hypotheses. In this case, the normalization constant would be identical for all hypotheses and therefore can be omitted when comparing hypotheses. However, the model parameters were trained for a different task, namely that of disambiguating alternative parse trees for the same word sequence.

It can be expected that parse trees from correct hypotheses and parse trees from incorrect hypotheses tend to exhibit distinct syntactic patterns. These patterns may be different from those relevant for disambiguation. Therefore, the disambiguation score is complemented with the set of all features that were used for disambiguation.

Table 4.10: Miscellaneous features.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dropdet[H]</code></td>
<td>omitting determiner from noun phrase with head H</td>
</tr>
<tr>
<td><code>H</code></td>
<td>singular noun</td>
</tr>
<tr>
<td></td>
<td>plural noun</td>
</tr>
<tr>
<td></td>
<td>nominalized adjective (strong declination)</td>
</tr>
<tr>
<td></td>
<td>nominalized adjective (strong declination, singular)</td>
</tr>
<tr>
<td></td>
<td>nominalized adjective (weak declination)</td>
</tr>
<tr>
<td></td>
<td>nominalized adjective (mixed declination)</td>
</tr>
<tr>
<td><code>dropdet-adja-n</code></td>
<td>omitting determiner from a noun phrase consisting of an adjective and a noun</td>
</tr>
<tr>
<td><code>main-clause[T]</code></td>
<td>main clause of type T</td>
</tr>
<tr>
<td><code>T</code></td>
<td>with Vorfeld</td>
</tr>
<tr>
<td></td>
<td>without Vorfeld</td>
</tr>
</tbody>
</table>

The two final sets of features combine information about partial parse trees with information that is specific to the speech recognition task. The feature template `partial-parse-trees-non-first[N]` is defined exactly as the `partial-parse-trees` template from Table 4.8, except that all feature values are zero for the first-best speech recognition hypothesis. These features penalize non-first-best hypotheses depending on their respective number of partial parse trees. This has the effect that a non-first-best hypothesis can be chosen as the new recognition result only if its “plausibility” exceeds that of the first-best hypothesis by a certain value. This margin can vary for different numbers of partial parse trees. The corresponding feature template is shown below:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>partial-parse-trees-non-first[N]</code></td>
<td>artificial parse tree with at least N partial parse trees and the rank of the hypothesis is 2 or higher</td>
</tr>
<tr>
<td><code>N</code></td>
<td>{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}</td>
</tr>
</tbody>
</table>

The prosodic features were mainly introduced to increase robustness with respect to out-of-grammar utterances and incorrect sentence boundaries. The basic idea is to penalize a partial parse tree less severely if it is adjacent to a prosodically marked phrase boundary. If, for example, the segmentation component misses a sentence boundary, this sentence boundary may still be indicated by prosodic cues. A hypothesis that correctly provides one partial parse tree for each of the two sentences may then be assigned a higher score. Similarly, if the word sequence of a hypothesis cannot be parsed as a whole, the parser might still provide some correct parse trees. If some of the parse trees are adjacent to prosodic phrase boundaries, this may support the given hypothesis.

The prosodic information is represented as a vector \((p_1, \ldots, p_n)\), where \(p_i\) is the probability of a phrase boundary occurring between the words \(w_i\) and \(w_{i+1}\) of a given hypothesis. The phrase boundary probabilities are approximated by the sentence boundary probabilities which are provided by the segmentation component (see next section). This approximation was judged to be reasonable after inspecting sentence boundary probabilities on development data. The resulting feature template is as follows:
4.5 Segmentation

The transcriptions produced by the present baseline speech recognizer already contain tags that indicate potential sentence boundaries. However, we are also interested in the actual sentence boundary probabilities between any two consecutive words. This information is used to compensate for incorrect sentence boundary decisions and to introduce prosodic information into the speech recognition hypothesis scores (see Section 4.4). As will be detailed below, this information is further used to split long sentence segments. For this reason, we have implemented a sentence boundary detection system along the lines of [LSS+06]. The algorithm and the exploited features will be briefly sketched in the following.

First, stylized pitch contours [SHWS98] are extracted from the speech signal. The speaker turns are detected with the algorithm of [BDF+00], and the fundamental frequency distribution within each speaker segment is approximated with a log-normal tied mixture model [SHWS97]. The first-best word segmentation of the speech recognizer provides the inter-word boundaries at which the relevant features are extracted.

The main features are the pause length, the presence of a speaker turn, the presence of a sentence boundary tag in the first-best transcription and several features based on the fundamental frequency. The fundamental frequency features indicate the slope of the fundamental frequency contour before the inter-word boundary, the fundamental frequency before the inter-word boundary and the difference of the fundamental frequency across the boundary. Rather than directly using the absolute frequency values, we estimate the probability that the current speaker produces a fundamental frequency that is at most as large as the observed value. These probabilities are modeled by the log-normal distribution mentioned above. The single non-acoustic feature is the sentence boundary probability estimated by a hidden event language model [SS96]. The model was trained on roughly 200 million words of newspaper text with the SRILM toolkit [Sto02].

As in [LSS+06], decision tree classifiers were trained to map the feature values to a posterior sentence boundary probability. The training was performed with the IND toolbox [BC92]. Bootstrap aggregating with 50 decision trees was used for smoothing. As the amount of training data was rather small (45 minutes of broadcast news data), we trained decision trees for different subsets of features. The posterior probabilities for the different features sets were combined in a maximum entropy model by means of cumulative binning [LSS+06, p. 1533].

The result of the above algorithm is a sentence boundary probability for each inter-word boundary of the first-best word segmentation. We assumed a sentence boundary at each position with a sentence boundary probability of more than 0.5. The resulting segments can still be too long for parsing. Thus, a segment is split at the position with the highest sentence boundary probability if it spans more than 25 words of the first-best segmentation. This heuristics assumes that positions with high sentence boundary probabilities tend to be “safe splitting points” in the sense that they separate fragments that are grammatical in isolation. Our experiments suggest that this is indeed the case: examining the positions with sentence boundary probabilities between 0.2 and 0.5, we observed that about 40% of these positions either marked a missed sentence boundary or separated a main clause from extraposed material. Other kinds of safe splitting points were not considered.

4.6 Discussion

In this chapter, we have proposed an approach of integrating formal grammar information into automatic speech recognition. The approach offers ways to deal with two problems pointed out in Section 3.6: the poor coverage of precise formal grammars and the grammaticality of many incorrect speech recognition hypotheses. Both problems are tackled by means of partial parsing and considering the plausibility of parse trees.

Novel aspects of the approach are the proposed robustness technique and the particular way of integrating syntactic information into hypotheses reranking. The latter aspect includes the use of the disambiguation score feature and the fragment features (with and without prosodic cues). The fragment features indicate, among others, whether a hypothesis is covered by the grammar or not. This information is already reflected
in the disambiguation score, but it can also explicitly be taken into account when reranking the hypotheses. Another novel aspect is the use of part-of-speech mismatch features for disambiguation.

The proposed approach is based on N-best reranking rather than lattice processing or even tighter integration into the speech recognition decoder. The reason for this is that our approach is intended for the integration of precision grammars. Processing with precision grammars is computationally expensive and requires a large amount of memory. One of the main advantages of lattice parsing is that common word sequences of hypotheses need to be parsed only once. This advantage cannot be exploited by precision grammars as it is infeasible to store all partial lattice parsing results in memory. Further, search techniques such as A* cannot be used as the parsing strategy for lexicalized precision grammars is typically not left-to-right. [CCP04] observed that the best results for speech recognition with statistical parsing models were obtained by means of N-best parsing. Several authors reported that lattice parsing led to gains in processing speed, but to a reduced or only marginally improved accuracy [Roa01a, HJ03, CCP04]. As our aim is not to provide an efficient algorithm but to investigate the benefit of linguistic knowledge, an N-best rescoring approach appears to be appropriate.

The most closely related work is that of [CRS05] and [MKHO06]. They used statistical parsers (the Collins parser and the Charniak parser, respectively) to derive parse trees for the N best speech recognition hypotheses. As statistical parsers cover arbitrary word sequences by design, robustness was not an issue. The hypotheses were reranked by means of features extracted from the parse trees. Both approaches used the averaged perceptron algorithm rather than conditional log-linear models. This is most likely due to the fact that they made extensive use of headword information. This results in a very large number of features and therefore calls for a powerful feature selection method as provided by the perceptron algorithm.

[CRS05] reranked the 1000 best hypotheses. Their features were constructed from general linguistic events such as sequences of words and part-of-speech tags, context-free rule applications, headword annotations of non-terminal symbols and head-to-head dependencies. They do not consider the probability of the parse tree. [MKHO06] reranked only the 20 best hypotheses but considered the 20 best parse trees for each hypothesis in order to average the feature values. Their feature set is richer than the one of [CRS05] and includes features for specific linguistic phenomena. They also use features derived from the probabilities of the parse trees.

[Beu07] used a formal grammar to parse and rescore speech recognition hypotheses. The score of a hypothesis was computed as a weighted sum of the acoustic likelihood, the language model score, the word insertion penalty and three features extracted from the best sequence of partial parse trees. The best sequence of partial parse trees was determined by means of a heuristics similar to the fewest-chunks criterion. The features counted the number of complete parses, the number of partial parse trees spanning two or more words and the number of single-word parse trees. The weights were optimized to minimize the empirical word error rate. In contrast to the proposed approach, the probability of a partial parse tree was not considered.
Chapter 5

Linguistic Subsystem

This chapter is mainly concerned with our German broad-coverage precision grammar and the related linguistic components. We will first give a brief introduction to HPSG, the grammar formalism that was adopted for this work. Then, we will discuss and motivate our particular choice of the formal devices that are used for expressing the linguistic constraints. The subsequent sections will present the grammar system, the parser and the German grammar that have been developed in the course of this thesis. In particular, we will present a novel contribution to HPSG parsing. Finally, we will describe and evaluate the part-of-speech tagger that was developed for tagging speech recognition hypotheses.

5.1 The Grammar Formalism

5.1.1 A Very Short Introduction to HPSG

In Head-driven Phrase Structure Grammar [PS87, PS94, SW99], linguistic objects such as words or phrases are modeled as typed feature structures. Figure 5.1 shows a simplified feature structure for the noun phrase “der Vogel” (the bird) in the attribute-value matrix (AVM) notation. A feature structure is denoted by a pair of square brackets and consists of a type identifier and a set of feature-value pairs. The feature structure shown in Figure 5.1 is of type phrasal-sign and has the features PHON, SYNSEM

and DTRS (the meaning of these features will be discussed later). Each attribute is assigned a value that is again a feature structure. Feature structures can also represent lists of feature structures. For notational convenience, lists are denoted as ⟨ ... ⟩.

The type of a feature structure defines the set of features that can be specified and the types of the values that can be assigned to each feature. Types are organized in an inheritance hierarchy, where subtypes inherit the feature declarations from their supertypes. Subtypes can substitute for their supertypes. For instance, the feature HEAD is restricted to take
values of type head. As noun and determiner are subtypes of head, feature structures of these types can also be assigned to the HEAD feature.

Indices such as \[ ] indicate structure sharing: all feature structure descriptions marked with the same index refer to the same feature structure. Thus, the value of DTRS|HD-DTR|SYNSEM|LOC|CAT|HEAD is identical to that of SYNSEM|LOC|CAT|HEAD in the example of Figure 5.1. Due to such reentrancies, the topology of a feature structure is a directed graph rather than a tree.

Besides offering feature structures as a means for describing linguistic objects, HPSG is also a theory on how linguistic information should be represented in feature structures. For example, phrases are described by feature structures of type phrasal-sign, whereas representations of words are of type lexical-sign. Both phrasal-sign and lexical-sign are subtypes of the more general type sign, which introduces the features PHON and SYNSEM. The value of the PHON feature was originally assumed to represent the sound of a word or phrase, but it is common to use an orthographic string instead. The SYNSEM feature contains the syntactic and semantic representations of the linguistic object. Feature structures of type phrasal-sign have an additional feature called DTRS (daughters). This feature embeds the daughter signs, i.e. the feature structures representing the constituents from which the given phrase (also referred to as the mother sign) is composed.

Linguistically speaking, the feature structure in Figure 5.1 describes a saturated nominative noun phrase composed of two daughter constituents, a head daughter (HD-DTR) and a non-head daughter (NHD-DTR). These constituents are words rather than phrases as their respective feature structures are of type lexical-sign. The head daughter is a nominative noun and the non-head daughter is a nominative singular masculine determiner that requires weak declension. The type of the feature structure embedded under DTRS specifies the kind of composition. In this example, a head constituent is combined with its specifier – or more precisely, a noun is combined with its determiner.

A head-driven phrase structure grammar is basically a set of constraints on feature structures. One set of constraints is implicit in the inheritance hierarchy and the feature declarations. Other constraints are represented as immediate dominance schemata, which are rough equivalents of grammar rules in other formalisms. In our example, the Head Specifier Schema is defined as follows:

[\text{SYNSEM|LOC|CAT|HEAD} \rightarrow \text{DTRS|HD-DTR|SYNSEM|LOC|CAT|HEAD}]

This constraint represents a generalization over a set of immediate dominance schemata in the sense that it applies to a feature structure if and only if a dominance schema from the set applies as well. This set consists of all immediate dominance schemata for which the type of the DTRS value is a subtype of headed-structure. This includes the Head Specifier Schema because head-specifier-structure is a subtype of headed-structure.

The Head Feature Principle captures the notion of a lexical head as introduced in Section 2.1. In any construction that involves a head daughter, the value of the head daughter’s HEAD feature is identical to that of the mother. This has the effect that the information encapsulated in the HEAD feature value (e.g. the part-of-speech) is percolated to all projections of a lexical head. In the present example, the Head Feature Principle ensures
that the noun phrase “der Vogel” inherits the part-of-speech (noun) and the case (nominative) of its head, the noun Vogel.

The final set of constraints is represented in the lexicon. The lexicon entries for our example are shown in Figure 5.2. Even for this toy example it can be seen that much grammar information is actually encoded in the lexicon entries. HPSG offers formal means for reducing the redundancy and capturing regularities in the lexicon, namely multiple inheritance hierarchies and lexical rules. Multiple inheritance hierarchies are used to combine linguistic information for different aspects of a word. Lexical rules derive one lexicon entry from another. An obvious application of lexical rules is to account for derivational morphology, e.g. the derivation of the adjective drinkable from the verb to drink.

Figure 5.2: Two example lexicon entries.
specific variants. We found that the particular engineering problem considered in this thesis requires a grammar formalism that facilitates efficient processing and offers a high flexibility in describing exceptional phenomena. Formal elegance, i.e. the use of a homogeneous set of simple formal concepts, was considered to be less important. As existing systems did not sufficiently meet these requirements, we decided to develop an own HPSG system and explore different design choices. The current system incorporates features from different existing implementations, namely PET [Cal01], LKB [Cop01], ALE/TRALE [CP03] and the Babel system [Mül96]. In the following, we will present an outline of the design space and the particular design of the current HPSG system.

As mentioned before, it is necessary to be specific about the actual properties of the formalism as there are different varieties of HPSG. Some variants let immediate dominance schemata constrain only the dependency structure of sentences, whereas the relative order of constituents is specified by a set of linear precedence rules [PS87, Chapter 7]. Sometimes it is assumed that an immediate dominance schema can account for an arbitrary number of non-head daughters, the actual number being specified by the head daughter [PS94]. Other variants allow for discontinuous constituents, which means that feature structures can represent discontinuous sequences of words [Rea96, Mül99]. Approaches with discontinuous constituents have particularly been advocated for formalizing languages with relatively free constituent order such as German.

The Babel system is one of the few systems that are based on discontinuous constituents. In PET, LKB, ALE and TRALE, constituents are assumed to be continuous and linear precedence is implicit in the immediate dominance schemata. All systems with the exception of ALE and TRALE require immediate dominance schemata to specify a fixed number of daughters.

Grammar systems further differ in what constraints on feature structures can be expressed. PET, LKB, ALE and TRALE allow for complex type constraints, i.e. implicational constraints of the form \( \tau \rightarrow FS \) for a type \( \tau \) and a feature structure \( FS \). TRALE also supports more general constraints of the form \( FS_1 \rightarrow FS_2 \), which are termed complex antecedent constraints. In ALE and TRALE, inequational constraints can be used to state that two feature structures are not identical with respect to structure-sharing. Finally, ALE, TRALE and the Babel system allow for the use of arbitrary relational constraints on feature structures. For example, the relational constraint \( \text{append}(A, B, AB) \) is fulfilled if and only if \( A, B \) and \( AB \) are feature-structure representations of lists and \( AB \) represents the concatenation of \( A \) and \( B \).

The formalism adopted in our grammar system assumes continuous constituents and immediate dominance schemata with a fixed number of daughters and implicit linear precedence. Our first experiments [BKP05a, BKP05b] were based on discontinuous constituents and a grammar along the lines of [Mül99]. However, it was found that this approach did not scale well to more difficult recognition tasks with higher lexical ambiguity and longer sentences. Discontinuous constituents have also been argued against on linguistic grounds [Mül05].

Unlike LKB, our system supports cyclic feature structures. Cyclic feature structures can occur naturally as a result of unifying acyclic feature structures, and we have found that it is difficult to anticipate or prevent situations where cycles can arise. Like [Car92, p. 35], we concluded that it is easier to allow cyclic feature structures than to exclude them.

As in PET, LKB, ALE and TRALE, the type system employs a closed-world semantics and the multiple inheritance hierarchy is assumed to be bounded complete, i.e. any two types in the hierarchy can have at most one most general common subtype [Car92]. However, there are no complex type constraints and no general complex antecedent constraints. Generalizations over feature structures in lexicon entries and immediate dominance schemata have to be captured by means of macro hierarchies rather than type constraints. Macros are parameterized feature structure templates that encapsulate specific linguistic information. Our notion of a macro is equivalent to that of a logical variable macro in the TRALE system. Complex antecedent constraints are restricted to feature structures of type phrasal-sign. This is sufficient to express principles that generalize over immediate dominance schemata, for example the Head Feature Principle introduced in Section 5.1.1.

The formalism allows to state relational constraints on feature structures. As in ALE and TRALE, the evaluation of a relational constraint is delayed if the argument feature structures are not sufficiently specified. Relational constraints can further be non-deterministic in the sense that they can be satisfied in several ways. For example, the relational constraint \( \text{delete}(L_1, X, L_2) \) is fulfilled if and only if \( L_1 \) and \( L_2 \) represent lists and \( L_2 \) is obtained by deleting the element \( X \) from \( L_1 \). If the argu-
ment L1 is specified but X and L2 are not, X could be any element from the list L1.

In summary, the implemented formalism employs only restricted types of implicational constraints, namely those that are necessary for stating immediate dominance schemas and principles. As was argued in [Cop01, Section 5.6], using only selected types of constraints and treating them differently in processing allows for efficient parsing. In this respect, the present formalism is similar to the PET and LKB systems. Unlike PET and LKB, our system allows to locally introduce arbitrary computations by means of relational constraints. Relational constraints can elegantly account for certain phenomena whose formalization in PET and LKB tends to be cumbersome or even inaccurate (e.g. argument attraction in verbal complexes, see Section 5.4).

The use of relational constraints is sometimes considered to be undesirable as an analysis that requires a simpler and less expressive formalism is generally preferable from a theoretic point of view. However, we argue that for an engineering problem as the one considered in this thesis, it is reasonable to introduce this additional flexibility. The use of relational constraints can lead to simpler solutions and – in some cases – even faster processing. For example, [Cry05] presented an account of relative clause extrapolation that involves matching an element in a list of feature structures. The original implementation on the LKB/PET formalism employed two dedicated grammar rules for iterating through the list and retrieving the matching element. Such an operation can be implemented more efficiently by means of relational constraints, particularly if the constraints are backed by procedural code (which is the case in the present HPSG system, see Section 5.3.1).

To conclude this section, we will briefly address an issue that is more related to parsing than to the actual grammar formalism. It is mentioned here because it also affects the way certain syntactic constructions are formalized. Some phenomena (e.g. the German main clause) are accounted for by assuming empty categories or traces. Empty categories roughly correspond to ε-productions in context free grammars: they are syntactic elements that cover an empty sequence of words. Empty categories typically represent a word that is overtly realized at some other part of the sentence. In most grammars, empty categories are initially underspecified and are (partly) unified with their overt counterparts at a later stage of the derivation. Empty categories can be explicitly represented in ALE and TRALE, but they can also be integrated into immediate dominance schemata as is done in grammars for PET and LKB. Irrespective of the actual representation, parsing with underspecified empty categories is expensive. As a consequence, grammars are sometimes artificially restricted in order to render processing feasible. Our grammar system offers so-called licenser rules as a means to specify empty categories and facilitate parsing without artificial restrictions. This technique will be discussed in Section 5.3.3.

5.2 The Grammar System

5.2.1 Architecture

An overview of our grammar system is shown in Figure 5.3. The linguistic information is contained in a set of lexica, a lexical grammar, a regular grammar, a phrasal grammar, the type hierarchy and a collection of macros. The macros encapsulate linguistic information that is used in the lexica and the grammars.

In order to parse a word sequence, each word is passed to the lexical processing component which applies the rules defined in the lexical grammar. Some of these rules are concerned with inflectional and derivational morphology and basically combine stems with affixes. Other rules are motivated by the grammar. In our grammar, for example, the feature structure of a finite auxiliary verb in sentence-initial position is completely different from that of the same finite verb in sentence-final position (see Section 5.4.2). The relation between the two feature structures is captured by a lexical rule that derives the former from the latter.

When a word is processed by the lexical processing component, a morphological analysis is performed to derive a feature structure for this word from the stem and affix representations contained in the lexica. The morphological analysis is an implementation of the algorithm presented in
besides stems and affixes, the lexica can also contain entries for fully inflected words. If full form entries for the given word exist, the respective feature structures are processed by the non-morphological rules only. The lexica can also include entries for multiword lexemes such as the German “nach wie vor” or the English “by and large”. Such expressions are identified in the word sequence and the corresponding feature structures are processed in the same way as full form lexicon entries. Unlike LKB, our system does not support the inflection of multiword lexemes.

The output of the lexical processing component is a set of lexical hypotheses. A lexical hypothesis consists of a feature structure as well as the start and the end position of the partial word sequence for which the feature structure is hypothesized.

It is also possible to generate a precompiled lexicon for a set of word forms by performing the lexicon processing for each word. The result is a mapping from a word form to a set of feature structures. This mapping can be stored on disk, and an index structure can be used to efficiently retrieve the precomputed feature structures on demand. Precompiling a lexicon saves both memory and processing time.

The regular grammar is used to model idiosyncratic expressions that follow very specific variation patterns. Examples are expressions of date such as “bis Ende August 2009” and “seit Mitte vergangener Woche”. Such patterns are rather cumbersome to handle within an HPSG grammar, but they can easily be captured with regular expressions. The regular grammar associates regular expressions with macros. If a regular expression can be matched with a subsequence of the input word sequence, the feature structure denoted by the macro is hypothesized for that subsequence. This is feasible for our application, as we are mostly concerned with the grammaticality of word sequences and do not need to know the internal structure of idiosyncratic expressions. The following regular grammar rule describes expressions such as “im September letzten Jahres”:

\[
@\text{lexicalized-adverbial}: \text{im} (\%\text{MONTH} | \%\text{SEASON}) \%\text{YEAR\_REL}
\]

The first term identifies the target macro. The terms starting with \% refer to other regular expressions that are to be embedded within the present expression.

The lexical hypotheses produced by the lexical processing component and the regular expression matcher are passed to the parser. The parser derives all complete and partial parse trees that conform to the rules and principles defined in the phrasal grammar. The resulting parse trees are provided as a packed parse forest. The parser will be discussed in greater detail in Section 5.3.

### 5.2.2 The Description Language

This section gives a glimpse of the description language in which macros, grammars and lexica are specified. It further discusses some of the features which simplify grammar development on the level of such descriptions.

The textual representations of macros, principles, grammar rules and lexicon entries are based on a description language which is very similar to the AVM notation introduced in Section 5.1.1. Examples of feature structure descriptions (as parts of macro definitions) are shown in Figure 5.4. Note that the types of feature structure descriptions do not have to be specified if they can be determined from the context. For example,
the type of the INHER feature value has not been specified as it immediately follows from the feature declarations in the type hierarchy. The index [local] indicates structure sharing between different parts of the feature structure and the argument which is passed to the macro. The header of the macro definition constrains this argument to be of type loc. The expression @topicalizable-synsem states that the constraints encapsulated in the macro topicalizable-synsem must also hold for the embedding feature structure. It is further possible to express disjunctions and conjunctions of feature structures.

Figure 5.4: Two example macro definitions.

Feature structure descriptions are compiled to a compact graph representation prior to the actual processing. The compilation of a feature structure description results in a (disjunctive) set of feature structures that do not contain embedded macros, disjunctions or conjunctions. In addition, the compiler infers all types that were not specified in the original feature structure description.

In order to facilitate the writing of lexical rules, the grammar system implements a framing mechanism along the lines of [Meu01]. In a lexical rule that derives a feature structure Y from a feature structure X, framing allows to specify Y only with the information that differs from X. All other information is automatically transferred from X to Y. Such a mechanism is important as only part of the syntactic information is changed in the course of inflection or derivation. Explicitly transferring the information by means of variables is tedious and error-prone in the face of grammar changes. Note that it is not trivial to formally capture the intuition of “differing information”. The algorithm proposed by [Meu01], which was also implemented in the TRALE system, represents one possible way to do so.

A final feature that simplifies the process of grammar development is the automatic generation of disjunction types. As shown in Figure 5.5, disjunctions of types can be represented in a type hierarchy. The use of disjunction types for features such as case, number and gender can significantly reduce the number of lexicon entries for certain words. This in turn improves the parsing efficiency with respect to both processing time and memory consumption. The grammar system automatically generates hierarchies of disjunction types if the respective base types are marked accordingly in the textual description of the type hierarchy.

5.3 The Parser

This section describes the Java HPSG parser that was implemented in the course of this thesis. The parser produces the parse trees which are used to extract the syntactic features for speech recognition. A description of the parsing algorithm is followed by a summary of the optimization techniques that were employed for efficient processing. Finally, we propose the use of licenser rules to avoid underspecified empty categories and we present a novel approach to parsing with such licenser rules.
5.3 The Parser

5.3.1 Parsing Algorithm

The basic parsing algorithm is bottom-up active chart parsing: all intermediate parsing results are stored as edges in a data structure called chart. Passive edges represent constituents that have already been derived from the input word sequence. Active edges correspond to rules for which some (but not all) of the right-hand-side arguments have been instantiated with previously derived constituents.

As noted in Section 5.1.1, the HPSG formalism models constituents as typed feature structures. A production rule $L \Rightarrow R_1 R_2 ... R_n$ represents an immediate dominance schema: $L$ corresponds to the mother feature structure and the $R_i$ to the daughter feature structures embedded under the feature dtrs. Instantiating a rule argument $R_k$ with a constituent (or passive edge) $C$ amounts to unifying the respective feature structures. Due to structure sharing between $L$ and the $R_i$, the instantiation may also specify information in $L$. If all rule arguments are successfully instantiated, the immediate dominance schema is satisfied and $L$ describes a valid constituent for the given part of the input word sequence. Thus, a passive edge for this constituent can be added to the chart.

[PS87, p. 144] introduced the Locality Principle stating that a mother feature structure must not have access to the internal constituent structure of its daughters. The Locality Principle entails that the dtrs value is only required in the left-hand-side feature structures of partially instantiated rules and can be omitted in the feature structures of passive edges. The removal of daughter information greatly reduces the size of feature structures.

The parser employs a key-driven parsing strategy [KK00, OC00a]. Key-driven parsing exploits the fact that parsing efficiency is affected by the order in which rule arguments are instantiated. In the HeadSpecifier Schema introduced on page 82, the non-head daughter is completely unspecified, whereas the head daughter is at least known to have a non-empty specifier list. As soon as the head daughter is instantiated, more information about the non-head daughter is available. Therefore, it is preferable to instantiate the head daughter first. The term key daughter refers to the most specific daughter, which depends on the immediate dominance schema and has to be indicated by the grammar developer. A key-driven parsing strategy first instantiates the rule argument that corresponds to the key daughter.

The only constraints that have been considered so far are immediate dominance schemata. However, the formalism specified in Section 5.1.2 also supports principles (implicational constraints on phrasal signs) and relational constraints. Principles are statically incorporated into the grammar rules whenever possible. Thus, if the antecedent $A$ of a principle $A \rightarrow C$ subsumes the left-hand-side feature structure of a grammar rule, the consequent $C$ is unified with that feature structure prior to parsing. If $A$ is consistent with the left-hand-side feature structure but does not subsume it, the constraint has to be checked at parse time for all feature structures that are produced by the respective rule. To this end, each rule is associated with a list containing all principles that have to be checked at parse time. When a passive edge is created by instantiating the final rule argument of an active edge, the resulting feature structure is unified with the consequents of all matching principles from the respective list.

A relational constraint is backed by a dedicated Java object that provides an evaluation method. This method may either specify some information in the relation arguments and then report a success or a failure, or it may delay the evaluation because the arguments are not sufficiently specified yet. In order to introduce non-determinism (i.e. multiple choice
The evaluation method can additionally specify a set of branching states which indicate alternative solutions. In the case of the relation \texttt{delete\(L_1, X, L_2\)} explained in Section 5.1.2, the branching states could simply be all possible element positions in the list \(L_1\), i.e. the potential positions of the element \(X\). Each branching state is later passed to the evaluation method which attempts to deterministically enforce the corresponding relational constraint. In the example of the \texttt{delete} constraint, one possible deterministic instance would be: “deleting the element \(X\) at position 3 of the list \(L_1\) results in the list \(L_2\)”.

Evaluations of relational constraints that are either delayed or not attempted yet are called \textit{partial evaluations}. Partial evaluations are represented as Java objects that are attached to active edges, passive edges and grammar rules. When a new edge is created, the partial evaluation objects from the involved edges and rules are duplicated. The duplicates are added to the new edge and the corresponding partial evaluations are executed in a round robin fashion. The execution terminates if one of the evaluations fails or if all evaluations either succeed or are delayed. This procedure is complicated by non-determinism. If an evaluation method returns a set of branching states, the new edge has to be duplicated for each branching state. As a result, the instantiation of a rule argument may result in two or more edges.

### 5.3.2 Optimizations

The parser has been optimized in several ways. First, unification is performed by means of the quasi-destructive graph unification algorithm with subgraph-sharing [Tom92, MCC00]. This algorithm is considered to be one of the most efficient graph unification algorithms for the application in natural language processing. An overview of alternative algorithms is presented in [Cal01]. The unification of types is based on lookup tables. This is feasible since our grammar employs a relatively small number of types (about 800 leaf types and 300 non-leaf types).

We have further implemented a number of optimization techniques proposed by [KKCM99], namely a precompiled rule filter, dynamic unification filtering and pruning of initial chart items. Finally, we use a stack-based memory management scheme for allocating feature structure nodes. This technique was proposed by [Cal01] and is implemented in the PET system.

The parser employs \textit{local ambiguity packing} [Tom85] in order to reduce the number of chart edges. With local ambiguity packing, an edge is not added to the chart if the chart already contains an edge that spans the same word sequence and is syntactically equivalent. Instead, the new edge is attached to the one that is present in the chart. If the daughter information is omitted as described in the previous section, syntactic equivalence can simply mean that the corresponding feature structures are identical. [OC00b] propose an approach to ambiguity packing that is based on subsumption rather than identity, i.e. an edge is attached to some other edge if it is less general than the latter. We did not implement this more complex approach as identity-based ambiguity packing worked sufficiently well and allowed for an efficient implementation based on hashing.

### 5.3.3 Licenser Rules

This section introduces licenser rules as a technique for avoiding underspecified empty categories. Licenser rules have originally been proposed in [Mül99] to account for partial verb phrase fronting in a grammar based on discontinuous constituents. As licenser rules may have non-adjacent rule arguments, they cannot straightforwardly be applied in grammar systems that are based on continuous constituents.

We propose the use of licenser rules in grammars with continuous constituents. To reduce the number of spurious chart edges, we extended the licenser rule concept with a technique which we call \textit{licenser binding}. We applied licenser rules and licenser binding to the analysis of the German main clause and observed significant improvements with respect to processing time and memory consumption compared to the alternative approach of using underspecified empty categories. In addition, our evaluations will show that the licenser rule approach can even lead to higher coverage because it obviates the need for the artificial constraints that are sometimes required by approaches based on underspecification.

We call two edges syntactically equivalent if they denote identical syntactic categories. Note that two edges can be syntactically equivalent even if they were derived in different ways and thus represent different syntactic structures.
5.3 The Parser

Problem Statement

Sentence (5.1) is an example of a German main clause:

\[(5.1) \text{gestern liess ihn sein Vater ausschlafen }\]
\[\text{yesterday let him his father sleep-late}\]
\[\text{‘yesterday, his father let him sleep late’}\]

In German main clauses, the verbal complex is split into a left and a right sentence bracket. The left sentence bracket contains the finite verb (liess in the above example) and the right sentence bracket contains all other verbal elements (ausschlafen). Each verbal element can contribute its own complements to the verbal complex, and these complements can be permuted almost freely between the two sentence brackets. In the above example, liess contributes the subject “sein Vater” and ausschlafen contributes the object ihn. To bridge the gap between the left and the right sentence bracket, it is common to assume an empty category (also called a trace in the present context) which acts as the sentence-final counterpart of its antecedent, the sentence-initial finite verb:

\[(5.2) \text{gestern liess ihm sein Vater ausschlafen } t_i\]

In this example, the symbol \(t_i\) represents the empty category (or more precisely: the empty verbal head) that corresponds to the verb liess. The dependency between these elements is indicated by the common index \(i\).

The empty verbal head allows the verbal complex “ausschlafen \(t_i\)” to be analyzed locally in the right sentence bracket. The verbal complex can then be combined with its complements and adjuncts, eventually yielding the verb phrase “ihn sein Vater ausschlafen \(t_i\)”. When the finite verb is combined with this phrase, it is checked whether the finite verb matches the assumed empty verbal head. A more detailed description of this particular account for the German main clause will be provided in Section 5.4.2.

Empty verbal heads pose a great challenge for bottom-up parsing. In actual implementations, empty verbal heads are typically underspecified because their antecedents are not known. In particular, the number and types of their complements are not sufficiently constrained. As a consequence, the parser hypothesizes many superfluous verb phrases that can never become part of a complete derivation. The same problems can occur with empty categories that are complements rather than heads. This is for example the case in the analysis of partial verb phrase fronting in German proposed by [Mühl05]. Licenser rules are, among others, a technique for locally introducing information about the antecedent of an empty category.

Licenser Rules

A licenser rule is a binary production rule whose right-hand side contains an argument marked as the licenser argument. In HPSG terminology, the licenser argument does not contribute to the phonological information of the mother sign. From the parser’s point of view, the application of a licenser rule results in a chart edge that spans exactly the same words as the non-licenser edge. The edges that instantiate the licenser argument and the non-licenser argument need not be adjacent. It can be specified whether the licenser edge has to be located before or after the non-licenser edge.

A licenser rule can be thought of as a unary rule that derives a new edge from the non-licenser edge. The licenser edge, which is not “physically” incorporated in the resulting edge, contributes information that is necessary for the derivation. The licenser argument can be used in two ways:

1. Information represented in the licenser feature structure prevents the resulting edge from being underspecified.
2. The presence of the licenser triggers the application of the unary rule. This can avoid unnecessary hypotheses if the resulting edge can only be part of a complete parse if there is a matching licenser.

An example for the former case is the trace-based analysis of the German main clause. In the parsing of sentence (5.1), a licenser rule can produce a chart edge for “ausschlafen \(t_i\)” by instantiating the non-licenser argument with ausschlafen and the licenser argument with liess. The licenser can be used to ensure that the feature structure for “ausschlafen \(t_i\)” is completely specified. The second case applies to complement extraposition. A more detailed discussion of how licenser rules are applied to those phenomena will be given in the Sections 5.4.2 and 5.4.5.

The parser processes licenser rules in a similar way to other grammar rules. In particular, it is possible to specify the key daughter of a licenser
rule (see Section 5.3.1). Upon the creation of a passive edge, the set of matching licenser rule arguments is determined by means of unification and the rule filter (see Section 5.3.2). This allows to efficiently match passive edges with active licenser edges.

**Licenser Binding**

We have extended the licenser rule concept with a *licenser binding* mechanism. Our basic assumption is the following: in a parse of a complete sentence, each edge serving as a licenser also has to appear as a non-licenser at some point of the derivation. More precisely: if a licenser rule produces an edge \( e \), the licenser edge has to appear as a sibling of some edge \( e' \) derived from \( e \).

It is possible to early reject edges which will never satisfy this requirement. Suppose that there are two edges \( e_1 \) and \( e_x \) such that \( e_x \) has been used as a licenser in the (bottom-up) derivation of \( e_1 \). If \( e_1 \) is combined with an edge \( e_2 \neq e_x \) and if \( e_2 \) and \( e_x \) overlap, no derivation of the resulting edge will be able to combine with \( e_x \). Therefore, two edges \( e_1 \) and \( e_2 \) may be combined only if the following *licenser binding constraint* holds:

\[
\text{For any edge } e_x \text{ that has instantiated a licenser argument in the derivation of } e_1, \text{ either } e_2 \text{ and } e_x \text{ do not overlap or } e_2 = e_x.
\]

Licenser binding can easily be implemented by adding a licenser set to each chart edge. For edges of lexical entries, the licenser set is empty. If two edges \( e_1 \) and \( e_2 \) with licenser sets \( L_1 \) and \( L_2 \) are combined by means of a non-licenser rule, the licenser set of the resulting edge is \( L_1 \cup L_2 \). If a licenser rule is applied and \( e_2 \) is the licenser edge, the resulting licenser set is \( L_1 \cup \{e_2\} \).

This simple variant of licenser binding has the disadvantage that it interferes with ambiguity packing: it may happen that two otherwise identical chart edges cannot be packed because they have different licenser sets. However, the above idea can be straightforwardly generalized to a variant which does not impair ambiguity packing. The basic idea is that a chart edge should carry a disjunction of licenser sets rather than a single licenser set. If two edges \( e_1 \) and \( e_2 \) with licenser set disjunctions \( L_{11} \lor \ldots \lor L_{1n} \) and \( L_{21} \lor \ldots \lor L_{2m} \) are combined by rule application, the disjunction of the resulting edge is \( (L_{11} \cup L_{21}) \lor (L_{11} \cup L_{22}) \lor \ldots \lor (L_{1n} \cup L_{2m}) \). If \( e_2 \) is packed onto \( e_1 \), the disjunction of the latter is extended to \( L_{11} \lor \ldots \lor L_{2m} \).

It now holds that a chart edge can be safely rejected as soon as the licenser binding constraint has been violated for each of its licenser sets at some point of the derivation.

In order to simplify the bookkeeping which is necessary for the above generalization, we actually use a more restricted variant of licenser binding. In general, we do not allow the packing of two edges with different licenser sets. The single exception are edges which were produced by the same licenser rule with the same non-licenser edge. The licenser sets of such edges will only differ with respect to a single element, namely the licenser edge of the preceding licenser rule application. This case is particularly interesting, as the packed edges can actually be ignored in the unpacking phase.

**Spurious Ambiguities**

A general problem arising from licenser rules are spurious ambiguities. The licenser is expected to take on a very specific role with respect to the sentence, each edge serving as a licenser also has to appear as a non-licenser at some point of the derivation. More precisely: if a licenser rule produces an edge \( e \) of the preceding licenser rule application. This case is particularly interesting, as the packed edges can actually be ignored in the unpacking phase.

The two instances of the auxiliary verb habe are syntactically and semantically identical. Therefore, the verbal complex gewusst \( t_j \) can be licensed by habe \( t_i \) even though habe \( t_j \) finally serves as the antecedent. As the “intended” licensing is also possible, we get one spurious ambiguity. Note that the licenser binding constraint is not violated: each licenser appears as a sibling of some phrase derived from a non-licenser. As mentioned in the previous section, spurious ambiguities of this kind can be reduced as a side-effect of ambiguity packing.

Still, spurious ambiguities are not banned completely. Consider the following situation. There are two chart edges \( e_1 \) and \( e_2 \) whose feature structures are unifiable, but neither feature structure subsumes the other. Each edge is used as the licenser of the same licenser rule with the same non-licenser edge. As a result, we get two edges \( e_{x1} \) and \( e_{x2} \) whose feature
structures incorporate information of their respective licenser. Because of this licenser information, neither edge subsumes the other. This in turn implies that identity-based and subsumption-based ambiguity packing does not apply to $e_{x1}$ and $e_{x2}$. As the licenser information of $e_{x1}$ and $e_{x2}$ is consistent with $e_1$ and $e_2$, both $e_1$ and $e_2$ can serve as the antecedent in derivations of both $e_{x1}$ and $e_{x2}$. Consequently, we get two spurious ambiguities in addition to the two proper readings.

The most general way to completely eliminate spurious ambiguities arising from licensing is to filter them out after parsing. This is achieved by “replaying” the unifications for each derivation tree without instantiating the licenser daughters. This operation yields a list of feature structures with fully instantiated daughters features. The spurious ambiguities are filtered out by removing the duplicates from this list.\[CO05\] use such a “replay pass” to reintroduce the semantic features which were removed prior to ambiguity packing. If such a device is already applied for other reasons, the above filtering procedure is relatively cheap.

**Evaluation**

In order to assess the benefit of the licenser rule approach, we performed experiments with two grammars that were derived from an earlier version of our grammar. The two grammars only differed in their account of German main clauses with right sentence bracket. The first grammar employed licenser rules whereas the second grammar made use of underspecified empty categories. In order to facilitate the processing with underspecified empty categories, the second grammar adopted a number of restrictions which were presented in [Cry03b] and which are implemented in the German DFKI grammar [MK00, Cry03a, Cry05].

For all experiments, the chart pruning technique by [KKCM99] was applied prior to parsing. This implies that a verb prefix was not contained in the chart unless it was selected by a verb. Without this optimization, the approach based on underspecification would have been at a serious disadvantage because it would have had to assume an empty verbal head for every potential prefix. The experiments and the results are detailed in [KP07]. In this section, we will only summarize the main results:

- The use of licenser rules and licenser binding reduced the parse time by a factor of 13.5. Comparable reductions were observed for the number of edges and the number of feature structure nodes, which are correlated with memory consumption. Further, it was found that no spurious chart edges were produced in this experiment.
- The licenser binding mechanism led to a substantial reduction of the parse time (-11%), the number of edges (-23%) and the number of feature structure nodes (-27%) compared to an approach with licenser rules only.
- The grammar with underspecified empty categories could not account for some sentences. The reason for this is that the mentioned restrictions have to make specific assumptions on the formation of verbal complexes in order to facilitate efficient parsing. For example, the maximum number of complements is restricted, and so is the number of complements that can be added or removed by an embedding verb. In [KP07], we have shown that these constraints can be inappropriate in certain situations.

In conclusion, we have successfully applied licenser rules to a grammar based on continuous constituents. The use of licenser rules has led to large improvements with respect to both processing efficiency and memory consumption, which are partly due to the proposed licenser binding mechanism. Moreover, our experiments suggest that licenser rules are also beneficial with respect to coverage. Licenser rules offer a direct way of dealing with the problem of underspecified empty categories without necessitating the introduction of artificial constraints.

**Related Work**

Approaches for processing empty categories more efficiently have been proposed in several publications. [JK94] were mainly concerned with the fact that an infinite number of empty categories or traces can be hypothesized at any position in the input sentence. They proposed to associate each lexical entry with a bounded number of traces. Each parse can consume
only those traces which are provided by the lexical items occurring in the sentence. Thus, the number of traces in any single parse is bounded and the parser is guaranteed to terminate (at least if the grammar does not permit infinite recursion). Besides demonstrating how traces can be assigned to lexical items in several government-binding analyses, they noted that lexical items could be used to partially specify their associated traces.

[Gei94] and [BFG+96] adopted a similar idea for the processing of German main clauses. Whenever a lexical item of a sentence-initial finite verb is accessed, the corresponding empty verbal head is made available to the parser. As this approach establishes the relation between the empty verbal head and its antecedent, the empty verbal head is fully specified.

However, the antecedent of an empty verbal element need not always be lexical. Counterexamples are fronted partial verb phrases and coordinated sentence-initial finite verbs in German:

(5.4) sie [suchte und fand], die Lösung $t_1$.

'she looked for and found the solution.'

The problem of underspecified empty categories also occurs for natural language generation. In the solution proposed by [SvPM90], the overtly realized antecedent can be thought of as being generated at the position of the empty category. Then, the antecedent is replaced by an empty element which is in turn specified according to the antecedent. This solution is related to the licensor rule approach in that it generates an empty category after its (phrasal or lexical) antecedent has been derived, incorporating all necessary information from the latter.

[KKCM99] proposed an approach for pruning lexical chart items. This approach is based on the observation that some chart items can only contribute to a complete parse if certain other lexical items are present as well. For example, split particle verbs such as “machte ... auf” (“opened”) in “er machte das Fenster auf” (“he opened the window”) typically involve a lexical item for the base verb (i.e. machte in the above example) that selects the appropriate prefix. This item can be omitted if the respective prefix is not available in the parser chart. This optimization technique is in fact a specialized licensing mechanism. As mentioned in Section 5.3.2, our parser employs this chart pruning technique in addition to the more general (but computationally more expensive) licensor rules.

5.4 The Grammar

The following overview of the grammar will be structured around groups of immediate dominance schemata. For the sake of brevity, we will not state the actual schemata but rather provide an informal description for each of them. Further, the schemata will be described in a procedural rather than a declarative way. It is hoped that this will make the grammar more comprehensible for readers that are not acquainted with HPSG. A more detailed outline of the grammar's coverage is provided in Appendix A.

The grammar integrates analyses of [Mül99], [Mül07], [Cry03a] and [Cry05]. Apart from the particular blend of these analyses, the main contribution of the author is the use of licensor rules in the analysis of the German main clause and complement extraposition. Another contribution consists in formalizing many rather specific constructions on the level of lexicon entries and domain-specific grammar rules. However, these constructions are beyond the scope of the general scheme presented here.

It should be noted that our grammar does not employ a semantic component, even though semantics traditionally is an important ingredient of HPSG. Thus, the grammar cannot be used to derive a semantic representation of a sentence, and it is not possible to generate a sentence from such a representation. Semantics was omitted as it was considered irrelevant for the task at hand.

5.4.1 Adjuncts and Complements

The following immediate dominance schemata are the most general ones used in the grammar. The Head-Complement Schemata combine a head constituent with one of its complements, for example a subordinator with a sentence as in “weil es regnet” or a postposition with a noun phrase as in “des Geldes wegen”. The complements that are required by a head constituent are represented in a complement list (also termed subcategorization or SUBCAT list). Further, a boolean feature INITIAL indicates whether the complements should precede or follow the head constituent. Note that verbal heads and coordinators require dedicated instances of the Head-Complement Schemata. These schemata will be introduced in Sections 5.4.2 and 5.4.4.
Head-Complement Schema (Initial): Combines a phrase-initial head with a constituent that conforms to the first element of the head’s complement list. The remaining elements are percolated to the complement list of the resulting phrase. The complement immediately follows the head. The head is not verbal and is not a coordinator.

Head-Complement Schema (Final): Combines a phrase-final head with a constituent that conforms to the first element of the head’s complement list. The remaining elements are percolated to the complement list of the resulting phrase. The complement immediately precedes the head. The head is not verbal and is not a coordinator.

The Head-Adjunct Schemata combine modifiers such as adjectives, adverbs or particles with appropriate head constituents. A modifier specifies a boolean feature PREMODIFIER that indicates whether it should precede or follow the head constituent. It further carries a description of the appropriate head constituents. This description is represented in the MOD feature.

Head-Adjunct Schema (Premodifier): Combines a premodifier with a head constituent that conforms to the MOD value of the premodifier. The premodifier immediately precedes the head constituent.

Head-Adjunct Schema (Postmodifier): Combines a postmodifier with a head constituent that conforms to the MOD value of the postmodifier. The postmodifier immediately follows the head constituent.

The above schemata are variants of the Head-Complement Schema and the Head-Adjunct Schema proposed in [PS94]. A roughly equivalent set of immediate dominance schemata is employed in the German DFKI grammar [MK00, Cry03a, Cry05].

5.4.2 Verbal Projections

A large part of the grammar is concerned with the construction of verbal complexes, verb phrases and sentences. The formal treatment of these constructions will be outlined in the following.

The Verbal Complex

The German verbal complex consists of the verbs in the right sentence bracket and – in main clauses – the finite verb in the left sentence bracket. In the following subordinate clause, the verbal complex is “erzählen wollte”:

\[(5.5) \text{dass er es erzählen wollte} \]
\[
\text{that he it to-tell wanted}
\]
\[
\text{'that he wanted to tell it’}
\]

Our account of verbal complex formation closely follows that of [Müll07]. It is assumed that a verb can select a verbal complement and attract (i.e. inherit) the complements of that verbal complement. Figure 5.6 shows a representation of the verb wollte. The value of the VCOMP feature specifies the verbal complement to be a sentence-final verb (or verbal complex) in base form\(^4\). The SUBCAT feature specifies the list of complements. This list is constrained to be the concatenation of the subject and the remaining complements of the verbal complement. The reason for this is that in German HPSG, the subject and the objects are typically represented in a single list for finite verbs and in separate lists for non-finite verbs.

Relational constraints enable a verb entry to arbitrarily alter the list of complements inherited from the embedded verbal complement. It is possible to remove complements (e.g. the subject in passive constructions), to add complements (e.g. the dative object for the control verb versprechen) or to replace complements (e.g. expressing the subject with a von prepositional phrase in passive constructions).

A complete verbal complex is represented by a feature structure with a fully specified complement list. These complements can then be combined with the verbal complex in arbitrary order. The formation of verbal complexes is captured by the following three immediate dominance schemata:

Head-Cluster Schema (Base): Combines a sentence-final verb with its verbal complement. The verbal complement immediately precedes the verb.

---

\(^4\)We do not follow [Müll07] in placing the verbal complement on the SUBCAT list but rather use a separate VCOMP list as in [Müll99]. This is due to technical reasons: it allows the rule filter (see Section 5.3.2) to distinguish between grammar rules for verbal complex formation and other rules.
### 5.4 The Grammar

**Figure 5.6:** A feature structure representation of the sentence-final finite modal verb “wollte”. Some features are omitted for clarity.

**Figure 5.7:** The empty verbal head corresponding to the overtly realized verb shown in Figure 5.6.

**Head-Cluster Schema (Flip):** Combines a sentence-final verb with its verbal complement. The verbal complement immediately follows the verb. This schema accounts for Oberfeldumstellung as in the sentence “dass er es hat erzählen müssen”, where the verbal complement of hat is “erzählen müssen”. As proposed by [HN89], Oberfeldumstellung is triggered by a FLIP + value specified in the verbal complement.

**Head-Cluster Schema (Trace):** Combines a finite sentence-final empty verbal head with its verbal complement.

The last schema calls for a more detailed explanation. As was noted in Section 5.3.3, a novel aspect of our grammar is the use of a licenser rule for implementing this schema. This will be explained using a variant of example (5.5):

(5.6) wollte, er es erzählen ti
wanted he it to-tell
‘did he want to tell it’

The empty verbal head \( t_i \) represents the sentence-final version of the sentence-initial \( \text{wollte}_i \). Following the analysis in [Miü05], the empty verbal head is represented as shown in Figure 5.7. Note that this representation is identical to the one of the overt sentence-final \( \text{wollte} \) shown in Figure 5.6, except for the empty PHON value and the recursive embedding of the local syntactic information (i.e. the LOC value) within the so-called DSL feature. The role of the DSL feature is to percolate this local syntactic information to all verb phrases that are derived from the empty verbal head, as will be explained in the following.

The empty verbal head can combine with its verbal complement according to the Head-Cluster Schema (Base). Then it can combine with its adjuncts and complements and eventually constitute the phrase “er es erzählen \( t_i \).” Because of the Head Feature Principle, this phrase carries the DSL value which mirrors the local syntactic information of the empty verbal head. The sentence-initial verb \( \text{wollte}_i \) is defined such that its single complement is a verbal phrase with an empty complement list and an appropriate DSL value. The verb requires the DSL value to match the DSL value of its corresponding empty verbal head. Thus, the verb can be
combined with the verbal phrase “er es erzählen t”, finally yielding the phrase “wollte er es erzählen t”.

Unfortunately, our parser cannot directly process empty categories. The usual way of introducing empty categories despite of this restriction is to create new grammar rules by “manually” instantiating rule arguments with empty categories. In our case, the head argument of the Head-Cluster Schema (Base) is instantiated with an empty verbal head. This yields a unary rule whose single argument represents the embedded verbal complex. As there are many possible empty verbal heads, it is common to use a single underspecified instance for this purpose.

It was demonstrated in Section 5.3.3 that this kind of underspecification is problematic with respect to processing efficiency. We therefore propose to use a licenser rule in order to avoid underspecification. This licenser rule adds a licenser argument to the unary rule constructed in the previous paragraph. The licenser argument is instantiated with the sentence-initial verb (in the above example wollte) at parse time, providing all information that is necessary to sufficiently specify the empty verbal head. Figure 5.8 shows the immediate dominance schema that corresponds to the resulting rule.

**Verb Phrases and Sentences**

Verb phrases and sentences with a sentence-final finite verb are derived by incrementally combining the verbal complex with adjuncts and complements. This is done by means of the Head-Complement Schema (Verbal/Final) below and the Head-Adjunct Schema (Premodifier) described in Section 5.4.1.

**Head-Complement Schema (Verbal/Final):** Combines the head daughter, a verbal constituent, with one of its complements. The complement immediately precedes the head.

This schema was proposed in [Mü05]. It employs the relational constraint delete(L1, X, L2) as defined in Section 5.1.2 to relate the complement list of the mother sign (L2) to that of the head daughter (L1). As X refers to the SYNSEM value of the complement daughter, the schema can be thought of as removing the given complement from the complement list when projecting the head daughter to the mother sign.

**Figure 5.8:** Head-Cluster Schema (Trace), a licenser schema combining an empty verbal head with its verbal complement. The non-licenser daughter represents the verbal complex that is selected by the empty verbal head. The licenser daughter is instantiated with a sentence-initial finite verb.
So far, it is possible to analyze sentences with a sentence-final finite verb, for example “(dass) er es erzählen wollte”. Sentences starting with a finite verb (e.g. “wollte er es erzählen?”) can be accounted for with the dsl approach introduced before. This approach requires a schema that combines the sentence-initial verb with its complement:

**Head-Complement Schema (Verbal/Initial):** Combines the head daughter, a sentence-initial verb, with one of its complements. The complement immediately follows the head.

In finite main clauses such as “er wollte es erzählen”, the finite verb is located at the second position, i.e. after the constituent that occupies the Vorfeld (prefield). The constituent in the Vorfeld can depend on embedded heads such as zu betrügen in the following example:

(5.7) [Um zwei Millionen Mark], soll er versucht haben, [eine Versicherung zu betrügen].

To formalize such non-local dependencies, we adopt the account of [PS94, Chapter 4]. In essence, the empty element t_i locally represents the constituent in the Vorfeld. The empty element introduces a non-local dependency by adding its relevant syntactic information to a dedicated list, the so-called slash list. This information is percolated in the derivation of the phrase “soll er versucht haben, eine Versicherung zu betrügen”. Finally, the Head-Filler Schema binds the element in the slash list with the constituent in the Vorfeld:

**Head-Filler Schema:** The head daughter, a sentence with a finite verb at the first position and a non-empty slash list, is combined with a matching filler constituent. The local syntactic information of the filler is unified with that of the slash element and the further percolation of this element is suppressed. The filler immediately precedes the head.

The remaining question is how the assumed empty elements are represented and processed. Here, HPSG offers a solution that does not require actual empty elements or traces that have a distinct position in the surface word sequence (see [PS94, Chapter 9.5] for a psycholinguistic argument in favour of this approach). The basic idea is again to manually instantiate grammar rules with empty categories. The following unary schemata for introducing non-local dependencies are based on the ones proposed in [Müller99]:

**Slash Introduction Schema (Complement):** Projects a head daughter with a single complement and an empty slash list to a phrase with an empty complement list and a slash list whose single element is the complement of the head daughter.

**Slash Introduction Schema (Adjunct):** Projects a head daughter with an empty complement list and an empty slash list to a phrase with a non-empty slash list. The single element on this list represents an adjunct modifying the head daughter. This accounts for modifiers in the Vorfeld as in “gestern wollte er es erzählen”.

In sentences like “ihr erzählen wollte er es gestern”, the Vorfeld contains a partial verb phrase. [Müller99] proposed an analysis of partial verb phrase fronting that makes use of licenser rules. We adopted this approach. As our grammar is based on continuous rather than discontinuous constituents, we obtained a slightly different set of schemata. The first schema applies to sentence-final verbs and accounts for sentences like “erzählen wird er es wohl dürfen”. The second schema introduces non-local dependencies for sentence-initial verbs as in “erzählen will er es”.

**PVP Slash Introduction Schema (Final):** Projects a head daughter with a verbal complement and an empty slash list to a phrase that represents the verbal complement on its slash list. The head daughter is a sentence-final verb and the licenser daughter matches the verbal complement.

**PVP Slash Introduction Schema (Initial):** This schema has the same effect as the previous one but applies to sentence-initial verbs. It takes into account that the verbal complement of a sentence-initial verb is represented within the dsl value of its single complement.

Up to now, we have only considered sentences for which at least one verb is placed in the right sentence bracket. For such sentences, it was assumed...
that an empty verbal head represents the finite verb in the right sentence bracket. If we extended this approach to main clauses with an empty right sentence bracket, the empty verbal head would consequently be the single element in the right sentence bracket. This would be problematic with respect to processing as the empty verbal head would have to be hypothesized at virtually every position to the right of the finite verb. So far, empty verbal heads were only hypothesized after a verbal complex.

As processing efficiency is crucial for performing real-world experiments, we resorted to a solution suggested by [Cry03a]. This approach assumes a left-branching structure for sentences with an empty right sentence bracket. This means that a sentence is constructed from left to right, starting from the finite verb. This approach is implemented by means of the Head-Complement Schema (Verbal/Initial) introduced above as well as two additional schemata. These two schemata are necessary because adverbials only modify sentence-final verbal phrases, i.e. verbal phrases that are marked with initial -.

**Head-Adjunct Schema (Verbal/Initial):** This schema is almost identical to the Head-Adjunct Schema. The main difference is that the mod value of the adjunct daughter has to match the head daughter except for the value of initial.

**Slash Introduction Schema (Adjunct/Initial):** This schema is almost identical to the slash Introduction Schema (Adjunct). The main difference is that the mod value of the adjunct description on the slash list does not specify the initial feature.

### Relative Clauses and Interrogative Clauses

Relative clauses are analyzed as proposed by [PS94]. A relative clause (e.g. “auf den er zusteuert”) is derived from a relative phrase (“auf den”) and a sentence with a non-empty slash list (“er zusteuert”). Relative phrases are marked by a non-empty rel list that contains the agreement information of the embedded relative pronoun. This information is introduced by the relative pronoun and percolated in the derivation of the relative phrase. We adopted the Relative Clause Schema from [Mül99]:

**Relative Clause Schema:** Combines a relative phrase with a sentence whose single slash list element matches the relative phrase. The sentence is headed by a sentence-final finite verb. The resulting phrase modifies a noun whose agreement features conform the those of the relative phrase.

Our account of interrogative clauses such as “warum er es erzählt hat” is similar to the above analysis, with the difference that an interrogative phrase is marked by a non-empty QUE list and the resulting interrogative clause is not a modifier:

**Interrogative Clause Schema:** Combines an interrogative phrase with a sentence whose single slash list element matches the interrogative phrase. The sentence is headed by a sentence-final finite verb.

### 5.4.3 Noun Phrases

This section describes the immediate dominance schemata that are dedicated to the formation of noun phrases. We follow [Mül07] in that all common nouns select a determiner that is represented in the noun’s specifier (spr) list. The HeadSpecifier Schema [PS94] is used to attach a determiner to a nominal projection:

**HeadSpecifier Schema:** Combines the head daughter with a non-head daughter that matches the first element of the head daughter’s spr list. The remaining elements are percolated to the spr list of the resulting phrase. The head daughter immediately follows the non-head daughter.

To account for noun phrases without a determiner, we adopt a unary schema that was employed in the grammar by [MK00]. This schema can be thought of as combining a nominal projection with an empty determiner:
5.4 The Grammar

**Empty Determiner Schema:** Projects a noun phrase with a non-empty spr list to a noun phrase with an empty spr list. The single element of the daughter’s spr list is unified with the description of an empty determiner that enforces strong declension.

Nouns with a mandatory determiner (e.g. singular count nouns) can prevent the application of this schema by marking the determiner description on their spr list accordingly.

As prenominal and postnominal modifiers are assumed to modify nominal projections with non-empty spr lists, they cannot modify pronouns. This excludes grammatical phrases such as “jemanden, den ich kenne” (someone I know). The following schema enables the postmodification of pronouns:

**Head-Modifier Schema (Pronoun):** Combines a pronoun with a postmodifier. The pronoun and the mod value of the modifier have to agree in their head information. The pronoun immediately precedes the modifier.

Nominalized adjectives as in “nur das medizinisch Notwendige” are derived by means of the following schema:

**Nominalized Adjective Schema:** Transforms an adjective phrase into a noun with identical agreement features.

The final schemata for the construction of noun phrases deal with prenominal genitives (“des Pudels Kern”), postnominal genitives (“die Stunde der Wahrheit”) and appositions (“Verteidigungsminister Scharping”).

**Prenominal Genitive Schema:** Transforms a genitive noun phrase to a determiner.

**Postnominal Genitive Schema:** Combines a nominal head with a genitive noun phrase. The genitive noun phrase immediately follows the head.

**Postnominal Genitive Schema (Pronoun):** Combines a pronominal head with a genitive noun phrase. The genitive noun phrase immediately follows the head.

**Apposition Schema:** Combines a nominal head with a proper name whose spr list is empty. The proper name immediately follows the head.

**Apposition Schema (Determiner Drop):** Combines a nominal head with a proper name whose spr list is non-empty. The proper name immediately follows the head. This schema allows for appositions with proper names that require a determiner in other contexts, e.g. “das Urlaubsland Türkei”

5.4.4 Coordination

Coordinators such as und in “der Mann und die Frau” are modeled as in the German DFKI grammar [MK00, Cry03a, Cry05]. A coordinator is assumed to have the right conjunct as its single complement. Further, its spec feature value represents the syntactic information of the left conjunct. The representations of the left and the right conjunct (i.e. the spec value and the subcat list element) are constrained to be identical in terms of structure-sharing, except for the semantic information.

For technical reasons, our grammar employs a dedicated instance of the Head-Complement Schema to combine a coordinator with its right conjunct. The following schema also applies to complements that are parts of verbal complexes as in “dass er es erzählen kann [und will]”. This is not possible with the other Head-Complement Schemata.

**Head-Complement Schema (Coordination):** Combines a coordinator with a syntactic element that conforms to the single element of the coordinator’s complement list. The coordinator immediately precedes that syntactic element.

The coordinator phrases that result from the above schema can be com-

---

6 For nouns and and noun phrases, the semantic information encapsulated within the cont feature also includes the number, gender and person features.
bined with a left conjunct by means of the Coordination Schema:

COORDINATION SCHEMA: Combines a coordinator phrase with a conjunct that matches the spec value of the coordinator phrase. The syntactic information of the resulting phrase corresponds to that of the conjunct and the spec value. The conjunct immediately precedes the coordinator phrase.

The following schema deals with correlative structures as in “weder der Mann noch die Frau”:

COORDINATION SCHEMA (CORRELATIVE): Combines two adjacent coordinator phrases with matching spec values. The two coordinators need to be compatible and their relative order has to be correct. The syntactic information of the resulting phrase corresponds to that of the spec values.

Finally, we employ a schema that allows for omitting determiners in coordinations, which frequently occurs in news reports. An example is “Ir
gendwie verlangen [Regierung und Opposition] ziemlich viel vom Volk”\textsuperscript{7}.

COORDINATION SCHEMA (DETERMINER DROP): Like the Coordination Schema, but the conjunct and the spec value do not have to agree on the value of the spr feature. The conjunct and the spec value describe nouns with non-empty spr lists, and at least one of them has a mandatory determiner. The syntactic information of the resulting phrase corresponds to that of the conjunct and the spec value, except for the empty spr list.

5.4.5 Extraposition

Extraposition refers to the phenomenon that a complement or modifier is separated from its head and “moved” to the right:

\begin{align*}
\text{(5.8) er wird } & \text{über das buch } t_i \text{ sprechen } [\text{das er schreibt}]_i \\
\text{he will talk about the book he’s writing’}
\end{align*}

For the extraposition of modifiers, we use an efficient approach proposed by [Cry05] which builds on work by [Kis03, Kis05]. In this approach, each head that could potentially be modified by an extraposed modifier introduces an anchor that carries the relevant linking information, e.g. the person, number and gender features for nouns. Anchors are percolated in the course of the derivation. The percolation mechanism is based on two lists that are represented in each constituent feature structure: a list of active anchors and a list of inert anchors. An anchor is called active if its corresponding head is followed by syntactic material of embedding constituents. For example, the anchor of buch is active in the phrase “über das buch sprechen” because of sprechen.

Active anchors enable the attachment of extraposed modifiers that match the anchor information. Our grammar employs three schemata for the attachment of extraposed modifiers:

ADJUNCT ATTACHMENT SCHEMA (NON-VERBAL): Combines a saturated, non-verbal head constituent with a modifier. The mod value of the modifier must conform to at least one active anchor of the head constituent. The modifier immediately follows the head.

ADJUNCT ATTACHMENT SCHEMA (VERBAL/FINITE): Combines a saturated, finite verbal head constituent with a modifier. The mod value of the modifier must conform to at least one active anchor of the head constituent. The modifier immediately follows the head.

ADJUNCT ATTACHMENT SCHEMA (VERBAL/NON-FINITE): Combines a saturated or unsaturated, non-finite verbal head constituent with a modifier. The mod value of the modifier must conform to at least one active anchor of the head constituent. The modifier immediately follows the head.

The last schema applies to fronted partial verbal complexes that are not saturated, i.e. that have a non-empty complement list. This allows to analyze sentences such as the following: “[Märchen $t_i$ erzählen, die Angst machen], möchte ich euch nicht”. Our grammar employs a relational constraint to search the list of active anchors for a matching anchor. This is in contrast to [Cry05] who made use of unary retrieval rules.
Like the anchors for modifier extraposition, extraposed complements are organized in an active list and an inert list. These lists are maintained by the same percolation mechanism and a similar set of schemata is used to attach extraposed complements:

**Complement Attachment Schema (Non-Verbal):** Combines a saturated, non-verbal head constituent with an extraposed complement. The complement must match an element in the list of active extraposed complements. The remaining elements are percolated to the resulting phrase. The complement immediately follows the head.

**Complement Attachment Schema (Verbal/Finite):** Combines a saturated, finite verbal head constituent with an extraposed complement. The complement must match an element in the list of active extraposed complements. The remaining elements are percolated to the resulting phrase. The complement immediately follows the head.

**Complement Attachment Schema (Verbal/Non-Finite):** A saturated or unsaturated, non-finite verbal head constituent is combined with an extraposed complement. The complement must match an element in the list of active extraposed complements. The remaining elements are percolated to the resulting phrase. The complement immediately follows the head.

Extraposed complements are introduced by a set of lexical rules. These rules produce lexical signs that differ in whether a particular complement is placed on the complement list or on a list of extraposed complements.

A novel aspect of our grammar is that some of these lexical rules are implemented as licenser rules. The licenser rules guarantee that a complement is only added to the list of extraposed complements if there exists a matching constituent somewhere to the right of the head constituent. This prevents the parser from producing constituents whose hypothetical extraposed complements will never be derived.

### 5.4.6 Subgrammars

The grammar contains 49 additional rules that make up the domain-specific subgrammars. These subgrammars describe certain types of expressions and account for the fact that the speech recognizer may split certain compound words (see Section 7.1). In particular, the subgrammars cover spelled-out numbers (“neunzehnhundreteinundachtzig”), expressions of date and time (“am zweiundzwanzigstenmaiviertausend eins”, “um viertelnachdrei”) as well as split compound nouns and acronyms⁸ (“zeitungsbericht”, “b. m. w.”). A detailed discussion of the subgrammars is beyond the scope of this thesis. However, their coverage is outlined in Appendix A.

### 5.5 The Part-of-Speech Tagger

As explained in Section 4.3.3, some of the features for parse disambiguation are based on the output of a part-of-speech tagger. A part-of-speech tagger for speech recognition hypotheses has to take into account the peculiarities of these hypotheses (see Sections 7.1 and 7.3). For example, compound nouns, acronyms and spelled numbers are written as separate words and the capitalization information provided by the n-gram language model is discarded in the course of N-best extraction. We have developed a part-of-speech tagger that is geared towards the processing of speech recognition hypotheses. Our experiments have shown that this part-of-speech tagger outperforms a retrained version of the TnT tagger [Bra00] on ASR-like text.

We have developed a maximum entropy part-of-speech tagger [Rat96] that is based on a conditional maximum entropy model and a cyclic dependency network [TKMS03]. The best tag sequence is determined by means of a beam search algorithm with a beam size of 100. The tagger model employs a set of general features and a set of features that are partly specific to the task of tagging speech recognition hypotheses. The general features are shown in Table 5.1. Features of this kind are used in virtually all part-of-speech taggers based on maximum entropy models.

⁸Note that frequent acronyms are also represented in the lexicon. Grammar rules for arbitrary acronyms are employed because the speech recognizer used in our experiments is not particularly good at recognizing arbitrary sequences of letters. It frequently occurs that the correct sequence does not appear in the N best speech recognition hypotheses at all. The acronym grammar can help to cope with incorrectly transcribed acronyms that are not listed in the lexicon. Yet, the disambiguation model may penalize the use of the acronym grammar rules.
The specific features include a set of features that deal with spoken numbers. For these features, the words occurring in numbers are partitioned into different classes. For example, one class contains only the word _und_, whereas another class contains the digits _zwei_, _drei_, ..., _neun_. Each feature indicates the presence of a sequence of two or three consecutive word classes, one of which corresponds to the word _w0_. For example, some feature is equal to 1 if and only if the word to be tagged is _und_ and the preceding word is a digit between two and nine. Other specific features are shown in Table 5.2. These features consult precomputed databases that were derived from 220M words of newspaper text. A couple of additional features are not listed here as they contributed very little to the overall performance.

The part-of-speech tagger was trained and evaluated on a transformed version of the German TIGER corpus [BDH+02]. The transformations included the removal of punctuation and the conversion to lower case. Further, numbers were spelled out and rare compound words were de-

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>a given list of personal names contains <em>w0</em></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>a given list of personal names contains <em>w</em> and <em>w</em></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>a given list of personal names contains <em>w</em> and <em>w</em></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td><em>w</em> occurs at least 10 times more often as upper case word</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td><em>w</em> occurs at least 10 times more often as lower case word</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td><em>w</em> and <em>w</em> occur more often as a single compound than as two consecutive words</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td><em>w</em> and <em>w</em> occur more often as a single compound than as two consecutive words</td>
</tr>
</tbody>
</table>

Table 5.2: The specific features for part-of-speech tagging. Each feature is described by the event it indicates.

compounded by means of the algorithm proposed by [Add03]. Finally, the original part-of-speech tags were mapped to the tagset of our part-of-speech tagger. This tagset is a reduced version of the Stuttgart-Tübingen Tagset [STST99]. It does not distinguish between different verb forms and employs a coarser classification of pronouns, but it contains additional tags for split parts of compound nouns and spelled numbers.

The corpus was divided into a training set (500K tokens), a development set (20K tokens) and a test set (80K tokens). The databases used by the features in Table 5.2 did not contain data from either of these sets. We trained our part-of-speech tagger and the HMM-based TnT tagger on the training set. The training of the maximum entropy model was implemented by Thomas Ewender on the basis of the reranking software presented in [CJ05]. After training, both taggers were evaluated on the test set. We observed a classification error rate of 4.7% for our part-of-speech tagger and an error rate of 6.9% for the TnT tagger.

5.6 Discussion

It was shown that the HPSG formalism lends itself to formalizing syntactic phenomena at a very abstract and general level, but also on the level
of idiosyncratic phrase structure rules and lexicon entries. This quality appears to be crucial for a real-world natural language processing task as considered in this thesis. This was one of the main reasons for adopting the HPSG formalism for this work, another reason being the large body of literature for German HPSG.

As there are several varieties of the HPSG formalism, we developed our own grammar system in order to be flexible with respect to different design choices. In fact, the first version of the system was based on discontinuous constituents. When it became apparent that this approach did not scale well to tasks with longer sentences and higher lexical ambiguity, the system and the grammar were adapted to a continuous constituency approach. As we particularly aimed at flexibility in expressing linguistic constraints, we added support for relational constraints. This is unusual for a system that is not based on PROLOG or LISP.

We have presented a novel contribution to HPSG parsing, namely the use of licenser rules in grammars with continuous constituents. In particular, we have applied licenser rules to the German main clause with right sentence bracket and to complement extraposition. We have further proposed a licenser binding mechanism to improve the efficiency of processing with licenser rules. We have demonstrated that licenser rules can significantly reduce both parsing time and memory consumption compared to the alternative approach of underspecified empty categories. Licenser binding has been shown to be beneficial as well.

We have provided a description of our large-coverage HPSG grammar. This grammar incorporates ideas from existing grammars that differ greatly in their use of formal devices: the Babel grammar [Müll99] which is based on discontinuous constituents, Stefan Müller’s TRALE grammar [Müll07] which makes heavy use of empty elements and relational constraints, as well as the DFKI grammar [MK00, Crys03a, Crys05] that employs neither of these. We believe that this confirms the flexibility of our approach.

Finally, we have presented a part-of-speech tagger that is tailored to the tagging of speech recognition hypotheses. It was shown that our tagger outperforms the TnT tagger [Bra00] on the task of tagging an ASR-like text corpus.
Chapter 6

Development of Linguistic Resources

This chapter describes the methods that were used to develop the linguistic resources for our experiments, with a main focus on the acquisition of lexical information. This aspect of our grammar-based approach to speech recognition is paid special attention because the use of lexicalized precision grammars in real-world tasks is only feasible if precise lexical information can be acquired in a methodical way. In addition to lexical acquisition, this chapter also touches on the related issue of identifying idiosyncratic constructions that are not covered by the grammar.

6.1 Introduction

The problem addressed in this chapter is the development of a lexicon and a grammar that sufficiently cover the sentences to be processed. These sentences are, of course, not known in advance. However, we assume to be given a set containing all words that can appear in these sentences. For our experiments, this set consists of the roughly 7000 words that occur in the experimental data (see Section 7.1).

With respect to lexicon development, we are particularly interested in two types of linguistic information. First, it is necessary to determine the syntactic properties of each open-class word (i.e. each noun, verb or adjective) in the given list of words. However, this is not sufficient as the lexicon also has to account for lexicalized expressions such as “nach wie vor” in German or “by and large” in English. It is therefore necessary to acquire such multiword expressions and adequately represent them in the lexicon.

In general, it is not possible to extract such information directly from dictionaries or other reference books. One reason for this is that precision grammars employ very specific and fine-grained distinctions of syntactic categories that are not captured by dictionaries. In addition, dictionaries tend to represent the prototypical usages of some word and neglect non-standard language use and the numerous variations that can be observed in corpus data. This problem is an expression of the variability of natural language, which has briefly been touched on in Section 2.4.

As a consequence, we decided to rely on data-driven, semi-supervised approaches whenever possible. However, we deliberately did not use treebanks or the output of statistical parsers as we didn’t want to weaken the argument that the proposed approach requires relatively little syntactically annotated data. The approaches discussed in the following sections are mainly based on a text corpus of 220 million words that was tagged with the TnT part-of-speech tagger [Bra00]. The corpus consists of news reports from the Frankfurter Rundschau that were published between 2nd January 1997 and 13th April 2002.

Besides lexicon development, this chapter is also concerned with identifying grammar errors and missing idiosyncratic constructions. This can be achieved by parsing corpus text and inspecting the parsing results. Such approaches will be discussed in Section 6.4.

6.2 Acquisition of Open-Class Words

Identifying Lexemes

Before the syntactic properties of the open-class words can be determined, it is necessary to identify the lexical units (i.e. the lexemes) that account for the given set of word forms. This was done by sending queries to the canoo.net morphological database [Can]. For each word form, the set of matching lexemes and the corresponding inflected word forms were retrieved (with kind permission of Canoo Engineering AG). The inflectional
class and the stem information of each lexeme was automatically inferred from the inflected word forms.

A special treatment was required for particle verbs. German particle verbs consist of two parts that can be separated in certain situations. An example is the verb *untergehen* (to sink): the prefix *unter* is separated in sentences like “das Schiff geht unter” (the ship is sinking) and attached in sentences like “dass das Schiff untergeht” (that the ship is sinking). Such particle verbs might not be explicitly listed in the given set of word forms.

As certain combinations of verbs and prefixes are very productive, we used a data-driven heuristics to identify particle verbs in the set of word forms. For each verb lexeme and each potential prefix, it was checked whether the respective participle and infinitive forms (*untergegangen* and *unterzugehen*) occur in the text corpus. If both forms existed, a new lexeme entry for the respective particle verb was constructed.

This approach does not work for particle verbs such as *anfreunden*, for which the verb *freunden* does not exist as a non-particle-verb and thus is not represented by a lexeme entry. In order to find such verb pairs, we applied a heuristics that is similar to the one described above. However, the participle and infinitive forms had to be constructed from the stems, which in turn were obtained by removing potential verb endings from candidate word forms. The resulting entries had to be verified manually.

All of the above processing resulted in a set of lexeme entries, each of which is annotated with complete morphological information. Subsequent processing steps attached additional annotations to the lexeme entries. This information was finally presented to the lexicon developer in order to help determining the syntactic properties.

**Marking Lexemes**

One type of annotation is the marking of lexeme entries that the grammar could derive from other lexemes. In particular, this is done for adjectival participles (*bedeutend*) and nominalized verbs (*Bemühren*). Such entries cannot simply be discarded, as the syntactic properties of the derived word may not be predictable from the original word. For example, *bedeutend* as in “eine bedeutender Fortschritt” cannot be derived from the verb *bedeuten*, as the latter always requires an object. Similarly, *Bemühren* allows for an infinitive clause complement as in “das Bemühren, etwas zu erreichen”, which (at least in the current grammar) cannot be derived from the verb *bemühren*. As a consequence, such lexemes are only marked, leaving it up to the lexicon developer to decide whether a dedicated lexicon entry is necessary or not.

**Collecting Evidence**

The next step is to automatically collect evidence in favor of the hypothesis that a given lexicon entry has a certain property. Table 6.1 shows the properties of common nouns that are distinguished in the present scheme.

Collecting evidence basically means to gather sentences from a text corpus that support a given hypothesis. The goal is that the lexicon developer can take a decision based on a small set of representative example sentences. This approach is presumably more efficient (and more accurate) than browsing through dictionaries, and the corpus information can be expected to be more reliable than the intuition of the lexicon developer.

We used two basic approaches for collecting example sentences. The first approach is to formulate one or more regular expressions that describe potential example sentences. The alphabet of the regular expression consists of symbols that represent disjoint sets of word forms and/or part-of-speech tags. A regular expression for the detection of infinitive clause complements of nouns is given below:

\[
\begin{align*}
( ( (\text{nN} .) | (, \text{igendet} ) \\
( \text{nN} \text{comma} \ (\text{nN}\text{igendet}\text{kon}\text{default})\ast ((\text{ptkzu vvinf})|\text{vvizu} ) ) \\
(\text{delimiter}|\text{comma}|\text{kon} )
\end{align*}
\]

The symbols appearing in the above expression are defined as shown in Table 6.2. The first *nN* symbol on the second line represents the noun for which an infinitive clause complement is detected. The symbol definitions ensure that *default* does not include certain words that introduce sentences (subordinate conjunctions and relative pronouns). As such words could be responsible for the occurrence of infinitive verbs, they are not allowed between the comma and the infinitive verb. The first line involves negations (1) and is used to exclude postnominal genitives as in “der Versuch der Regierung, diese Waffen zu verbieten”.

This regular expression is very restrictive. The aim is to find only sentences that are very likely to be good examples for the given property. Many perfectly valid examples are ignored, but it is hoped that the corpus is large enough to contain at least one example of this particular form,
Table 6.1: The properties of common nouns with illustrating excerpts from the text corpus. Each property is either present or not. In the case of a mass noun, the sentential complements can be specified differently for singular nouns with and without determiner.

<table>
<thead>
<tr>
<th>Type</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>mass noun</td>
<td>Ich hatte mehr Angst vor Kjetil-Andre Aamodt (...)</td>
</tr>
<tr>
<td>temporal noun</td>
<td>Erst acht Monate später sollte er zurückkehren (...)</td>
</tr>
<tr>
<td>quantity noun</td>
<td>Ich kaufte zwei Dosen Bier (...)</td>
</tr>
<tr>
<td>title noun</td>
<td>Meister Friedensreich Hundertwasser (...)</td>
</tr>
</tbody>
</table>

Table 6.2: The symbols used in the regular expression for noun phrases with infinitive clause complement. The above definitions assign a unique symbol to each word/tag pair. The part-of-speech tags are those defined in the Stuttgart-Tübingen Tagset [STST99]. The EXCLUDE symbol prevents subordinate conjunctions, relative pronouns and finite verbs from being included in the DEFAULT symbol.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMA</td>
<td>tag = $,</td>
</tr>
<tr>
<td>DELIMITER</td>
<td>tag ∈ {$, $(, $)</td>
</tr>
<tr>
<td>GENDET</td>
<td>word ∈ {einer, eines, der, des, jener, jenes, dieser, dieses, meiner, meines, seiner, seinen, ihres, ihres}</td>
</tr>
<tr>
<td>KON</td>
<td>tag = KON</td>
</tr>
<tr>
<td>NN</td>
<td>tag = NN</td>
</tr>
<tr>
<td>PTKVZU</td>
<td>tag = PTKVZU</td>
</tr>
<tr>
<td>VVINF</td>
<td>tag = VVINF</td>
</tr>
<tr>
<td>VVIZU</td>
<td>tag = VVIZU</td>
</tr>
<tr>
<td>EXCLUDE</td>
<td>tag ∈ {KOUI, PRELS, PRELAT, VAFIN, VMFIN, VVFIN}</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>else</td>
</tr>
</tbody>
</table>

The basic idea is to train a statistical model \( P(\text{property} | \hat{f}(l)) \) which estimates the probability that a lexeme \( l \) has a certain syntactic property, given a set of features \( \hat{f}(l) = (f_1(l)...f_N(l)) \) extracted from a text corpus. A feature \( f_i(l) \) essentially searches the corpus and counts how often the

provided that the property holds for the lexeme. The use of restrictive patterns is characteristic for the approach based on regular expressions.

Regular expressions have shown to work poorly for some properties. For example, the absence of a determiner is not a good indicator for a mass noun, as determiners of count nouns can also be omitted in appropriate contexts such as headlines. The actual word context can be much more informative. For example, a noun is likely to be a mass noun if the preceding word is viel, etwas or bisschen. Our second approach attempts to find representative example sentences by considering a set of less specific features. These features may include the presence of certain word contexts, but also regular expression matches.
lexeme \(l\) occurs in a specific context. It was also tried to use relative frequencies in addition to the raw counts, but these additional features did not improve the overall accuracy. The training of the model requires an initial lexicon, which in our case was available from earlier experiments.

This model is not primarily used to compute the probability that \(l\) has a certain property, although this information may also be helpful for the lexicon developer. The actual use is based on the fact that the model encodes the relative importance of the different contexts, or rather of the context features \(f_i(l)\). For a sentence containing the lexeme \(l\), the features \(f_i(l)\) are set to \(\infty\) if their associated contexts are present, and zero otherwise\(^1\). The resulting probability \(P(\text{property} \mid \hat{f}(l))\) can be interpreted as a measure of how strongly the sentence supports the hypothesis that \(l\) has the given property. This measure of relevance is computed for all potential example sentences in the corpus and the \(N\) most relevant examples are presented to the lexicon developer.

\(^{1}\)The feature is set to \(\infty\) rather than \(1\) because the model is based on raw counts. If some context is positively correlated with the given property, the fact that the context appears only once in a large text corpus can imply that the property does not hold.

\[\text{[BB03, Bal05b, Bal07]}\] explored the use of statistical models for the task of automatic lexical acquisition. We adopted their approach of using pattern occurrence counts as features and applied it to the problem of weighting example sentences. In contrast to their work, we used a maximum entropy model with cumulative binning \[\text{[LSS'06, p. 1533]}\] rather than the TiMBL memory-based learner \[\text{[DZv03]}\]. We also experimented with TiMBL but found that the maximum entropy approach achieved a higher classification accuracy.

Although this approach could be used with very general feature templates, we employed small sets of relatively specific, linguistically motivated features. For example, the mass noun property was predicted from the preceding word and the preceding part-of-speech tag. Main clause complements were identified by means of a regular expression that tends to accept many incorrect examples. Thus, the classifier-based approach was used to properly weight the example sentences. The features indicated whether a certain word precedes the noun, or whether a certain word is present in the Vorfeld or the left sentence bracket of the main clause. This allowed the classifier to capture the tendency that main clause complements correlate with definite articles and that they typically start with a pronoun and an auxiliary verb in subjunctive mood. Similarly, the weights of potential examples for interrogative clause complements depended on the form of the interrogative word.

**Proper Nouns**

The classifier-based approach is also used for the acquisition of personal names and geographical names. However, the general scheme is slightly different for this task. As proper nouns are not well covered by the Canoo database, the classifiers are applied to word forms rather than lexeme entries. The lexicon developer thus has to identify proper nouns in the given set of words rather than specifying their syntactic properties.

First names and last names are modeled separately using similar sets of features. Among others, these features indicate whether other proper nouns (i.e. NE tags) or titles are present in the immediate context of the word, or whether the word occurs in certain patterns that are typical for appositions, direct speech or indirect speech. Examples of such patterns are the tag sequences \[\text{[$, NE NE $,]}\] and \[\text{[$, VVFIN NE NE $.]}\].

The detection of geographical names is based on two groups of simple features. The features of the first group determine if the word in question is tagged as a proper noun, preceded by the preposition in, nach or aus, and followed by a non-noun. The second group contains features that indicate patterns such as “in \(X\) geboren” and “in \(X\) lebende”. Despite of their simplicity, the above features have shown to work well for both classifying word forms and weighting examples.

**Discussion**

We have outlined our general procedure for the acquisition of open-class words and we have presented an approach for assisting the lexicon developer by automatically extracting example sentences from a text corpus. We will next briefly comment on the practical use of this approach in the context of our experiments.

Extracted example sentences have shown to be very useful for the acquisition of nouns. In fact, it was possible to define suitable patterns and features for all of the relevant properties. The use of example sentences allowed to enter the roughly 2200 common nouns and 1800 proper nouns from our experimental data in about two weeks’ time. We think that the success of this method is due to the fact that verifying the validity
of an example sentence is more efficient and reliable than consulting the linguistic intuition and making up supporting examples. Further, the examples can be structured around the actual syntactic properties that are to be determined. This is in contrast to dictionary information, which is typically organized according to different criteria.

Determining the syntactic information of verbs (which is essentially the valence information) was much more involved. This information was mainly obtained from the Duden dictionary [Dud99]. The automatic extraction of example sentences was only feasible for the purpose of detecting certain types of complements. In particular, it was applied to prepositional objects, sentential complements and pronominal adverbs substituting for extraposed sentential complements (e.g. “darauf, dass ...”). Further, it helped to determine whether a verb is likely to be used with a dativeus commodi/incommodi and whether a verb allows for the so-called coherent construction.

The acquisition of adjectives is less critical because there are only a few of them (about 500 in the present lexicon) and most of them do not have any complements. Automatically extracted example sentences were used to indicate adverbial and predicative uses of an adjective. Further, they ensured that occasional complements did not go unnoticed. The considered complement types were prepositional objects and pronominal adverbials substituting for extraposed sentential complements. Other complement types (most prominently dative objects) were entered by intuition.

It was observed that the current approach tends to select example sentences that are very similar. It is an open question how more diverse sets of examples could be produced.

**Related Work**

Most related work is aimed at automatic lexical acquisition. [Bal05a] used a set of binary features which are extracted from the context of a target word. The feature values were aggregated over a large text corpus, yielding a raw count and a relative frequency for each feature. The aggregated feature values were used to classify the target word. The training of the classifier required an initial lexicon.

In supertagging [BJ99], a classifier is used to predict the precise syntactic properties of a target word, given the textual context of the word.

A possible application of supertagging is to dynamically create a lexicon entry whenever the parser encounters a word that is not contained in the lexicon. [ZKV106] and [DKN08] used this technique for dealing with unknown words in HPSG parsing. The idea of representing the most frequent words in a lexicon and dynamically predicting the syntactic properties of the remaining words could also be applied to the parsing of speech recognition hypotheses. A drawback of such an approach is that a supertagger can adapt to errors in the recognition hypotheses. Thus, the grammar-based language model would be less effective in penalizing ungrammatical hypotheses.

[Bal05b] and [NKZ+08] used a supertagger to predict the syntactic properties of each occurrence of an unknown word in a text corpus. The resulting information was used to automatically create lexicon entries for the unknown words. This approach of extending the lexicon is applicable to the speech recognition scenario because the supertagger is applied to grammatical text. However, the training of a supertagger requires a syntactically annotated text corpus. As noted in Section 6.1, we decided not to use such resources.

[ZBK07] reported that parsing accuracy is much more affected by missing lexical information than by additional incorrect information (e.g. additional readings of verbs or nouns). The reason for this is that the disambiguation model can cope well enough with the increased lexical ambiguity, whereas a missing reading may prevent a sentence from being analyzed altogether. However, these results cannot necessarily be carried over to the task of reranking speech recognition hypotheses, where precise lexical information can help to disambiguate between correct and incorrect hypotheses. Thus, fully automatic lexical acquisition may be harder for the present task.

### 6.3 Multiword Expressions

[SBB+02] classified multiword expressions into *lexicalized phrases* and *institutionalized phrases*. Institutionalized phrases reflect strong preferences with respect to the choice of words and are more relevant for the generation of text from semantic representations. In the present context, we are mainly concerned with lexicalized phrases, which can be described as
phrases whose syntactic or semantic properties cannot be derived from their parts. In fact, we are only interested in syntactic compositionality. For example, the expression “den Löffel abgeben” (to kick the bucket) has a very specific meaning that cannot be derived from its parts. From a syntactic point of view, however, this expression is simply a transitive use of the verb abgeben and thus has not to be accounted for explicitly. This is in contrast to the expression “Schlange stehen” (to queue up), where Schlange cannot be analyzed as a noun phrase due to the missing determiner and stehen generally does not allow for an accusative object.

To be more precise, we consider expressions whose syntactic properties cannot be derived from its parts by the given grammar. These include expressions like “In- und Ausland”, which have to be regarded as compositional with respect to both syntax and semantics. Our notion of multiword expressions is also influenced by the actual parsing task. For example, idiosyncratic compound nouns like Kopf-an-Kopf-Rennen are treated as multiword expressions because the speech recognizer transcribes them as separate words.

The remainder of this section will first describe the types of multiword expressions that were considered in our experiments. Finally, our particular procedure for extracting multiword expressions will be outlined.

Types of Multiword Expressions

Table 6.3 shows the types of multiword expressions that are considered in the present scheme. The lexicon entries for multiword adverbials, nouns and adjectives correspond to those of their single-word counterparts, with the only difference that their morphological information is specified in terms of multiword full forms. Fixed adverbial prepositional phrases like “auf alle Fälle” are simply added as multiword adverbials. Note that this is not necessary if the given combination of preposition and case already allows for adverbial use, as e.g. in “auf allen Vieren”. Special lexicon entries are required for stereotypical phrases like “alles Unsinn” and plural noun phrases such as “Bund und Berlin”.

The multiword expressions discussed in the previous paragraph were clusters of words that did not undergo any variation besides inflection. We also account for certain types of multiword expressions with a variable internal structure. Expressions like “in Anbetracht + NPgenitive” are modeled as preposition-like elements that select for a noun phrase with a certain case. Further, there are partially fixed prepositional phrases of the form “auf ... Weise” that exhibit irregular syntactic properties (in this example the adverbial use and the optional determiner). This type of multiword expression is handled by defining a special preposition that selects a noun phrase with a specific head noun.

We further account for light verb constructions with fixed lexicalized objects, such as “etwas in Kauf nehmen”. This is done by means of a dedicated lexicon entry in which the light verb selects for the lexicalized object, which is treated as a special kind of verb prefix. In some cases, combinations between fixed prepositional phrases and light verbs are productive. As this is often the case for motion verbs, it is possible to define lexicalized prepositional phrases as directional adverbials that can be selected by such verbs.
6.3 Multiword Expressions

Potential multiword expressions were extracted in the standard way: suffix arrays [MM90a, YC01] were used to identify all tagged word sequences that occurred at least 5 times in the corpus and that were composed of at most 10 words from the given set of word forms. For each sequence, it was determined how strongly the word/tag pairs were associated with each other. Pointwise mutual information [Fan61, CH90] was chosen as the association measure for word pairs. For sequences with more than two words, the association strength was computed with the approximation proposed in [SJ01].

The tagged word sequences were sorted in decreasing order of association strength. Next, they were manually inspected starting from the top-ranked word sequence. Each word sequence that was considered to be a relevant multiword expression was marked and later added to the lexicon. The manual inspection was stopped when the event of a relevant multiword expression occurring became too rare.

This general approach was repeated for different types of word sequences. For example, word sequences with exactly two words were considered separately as the mutual information measure is exact in this case. Further, manual inspection was performed for those word sequences that contained unknown words, i.e. words that were not covered by the lexicon. For example, the unknown word \textit{air} was found to occur in the multiword expressions \textquotedbl air canada\textquotedbl, \textquotedbl air china\textquotedbl and \textquotedbl air force one\textquotedbl. Finally, light verb constructions with prepositional objects were identified from word sequences with a particular part-of-speech pattern, namely a preposition followed by an arbitrary word and a verb.

Acronyms were automatically identified and added to the lexicon. The identification was based on a simple heuristics, namely that of collecting all words that are tagged as common or proper nouns and that start and end with an upper case letter. The number and gender of an acronym were predicted by a maximum entropy classifier. The features indicated whether a certain form of a definite or indefinite article was observed in front of the acronym, and whether it was among the two most frequent forms for that acronym. This simple classifier was trained on data from the TIGER corpus [BDH*02] and turned out to work very well.

Multiword expressions were not exclusively acquired by means of the above methods. In fact, parsing-based approaches (which are discussed in the next section) have shown to be a rich source of multiword expressions. These approaches are more general in that they also allow to detect errors in the grammar and the lexicon.

6.4 Parsing-based Approaches

The approaches discussed in this section aim at enhancing the lexicon and the grammar by analyzing parser output. These approaches are related to the error mining technique proposed by [van04]. The basic idea of error mining is to parse a large text corpus and to determine the set of sentences for which at least one complete parse could be found. Next, it is computed how often a certain word or word sequence occurs in parsable sentences, relative to the total number of occurrences. Words or word sequences for which this value is small can indicate specific shortcomings of the grammar or the lexicon.

We followed the idea of parsing large amounts of data and analyzing the parsability results. However, as the given lexicon was rather small, we attempted to parse isolated constituents rather than sentences. Further, the unparsable phrases were examined manually. This was facilitated by restricting the manual inspection to phrases with a relatively high degree of association with respect to the pointwise mutual information measure (see Section 6.3).

In order to identify isolated constituents, we exploited the fact that the Vorfeld of the German main clause is typically occupied by a single constituent. The Vorfeld generally covers the part to the left of the sentence-initial finite verb, though it may be framed by coordinating conjunctions like denn and adverbial connectors such as hingegen. Potential instances of the Vorfeld were extracted in a similar way to multiword expressions. Suffix arrays were used to extract all word sequences that occurred at least 10 times in the corpus and that were not longer than 10 words. Next, we selected those sequences which started with the generic sentence boundary marker and ended with the finite verb marker. Again, word sequences were only considered if they were composed of words from the given set, including the two marker symbols.

Each Vorfeld word sequence was parsed after being converted to a format that resembles the speech recognizer hypotheses. Personal names and geographical names that are homonyms of other words were removed.
from the lexicon as they often lead to undesirable analyses. For example, the idiosyncratic expression “über kurz oder lang” would be parsable if kurz and lang were interpreted as personal names rather than adjectives. A word sequence was considered to be parsable if it could be analyzed as a noun phrase, a prepositional phrase, an adverbial, a directional complement or a predicative adjective.

All unparsable word sequences with a sufficiently high association value were inspected manually. This yielded many multiword expressions but also grammar errors and missing syntactic constructions. Examples of missing constructions are the postnominal selbst as in “er selbst” or various expressions of date that were later incorporated in the regular grammar, e.g. “im september letzten jahres”.

Parsable word sequences that could be analyzed as a prepositional phrase but not as an adverbial were checked as well. If such a word sequence was found to allow for adverbial use (e.g. “auf alle fälle”), it was entered as a multiword lexeme.

A similar parsing-based approach was applied to single words. The text corpus was searched for words that could be decomposed into elements from the given set of word forms. These words were normalized to resemble the speech recognition hypotheses: hyphens were replaced by spaces, acronyms were written as letter sequences and the resulting string was converted to lower case. Additional decompounding was not necessary as the lexical grammar can analyze arbitrary noun-noun compounds. The normalized words were parsed and the unparsable words were examined. As a result of this process, compound nouns such as “dritte welt länder” and “s. p. d. geführte” were identified and entered as multiword lexemes.

6.5 Discussion

We have outlined our scheme for the development of the lexicon and – to a lesser extent – the grammar. This scheme consists of a variety of methods for determining the syntactic properties of lexical units, for identifying the relevant lexical units and for detecting shortcomings of the grammar and the lexicon. Besides the morphological database on canoo.net, the main source of linguistic information was a part-of-speech-tagged text corpus that seemed to be sufficiently close to the given target domain. Dictionaries were only consulted for the task of specifying verb valence information.

The described methods were used to create a lexicon that covers the 7000 word forms occurring in our experimental data. The resulting lexicon contains about 600 adjectives, 4000 nouns and 3300 verbs. The verbs are annotated with more than 11 000 valency frames from a set of about 880 different frame types. The number of multiword lexemes amounts to 1200, not including the 1400 acronym entries that were generated automatically. The lexicon entries for the open-class words give rise to about 120 000 inflected forms.
Chapter 7

Experiments

This chapter describes experiments on a broadcast news transcription task and presents the results. It is further investigated how the speech recognition performance is affected by different factors such as the difficulty of the speech recognition task and the available linguistic information. In addition, different characteristics of the approach and the involved components are evaluated.

7.1 Baseline System and Data

Our experiments are based on word lattice output of the 300k LIMSI German broadcast news transcription system [MA03, GLA02]. The speech recognizer is a multi-pass system that employs continuous density hidden Markov models with Gaussian mixtures for acoustic modeling and 4-gram backoff language models. The vocabulary size is 300,000 words.

The lattices cover 6 broadcasts of the German news show Tagesschau. The data of three news shows was used in preliminary experiments and for the training of the segmentation component. The experiments reported in this chapter are based on the remaining three news shows (broadcast on the 29th of April, the 15th of May and the 23rd of May 2002). We will subsequently refer to the latter three news shows, unless otherwise noted.

The data used in our experiments amounts to 603 sentences, 7477 words, or a signal length of 52 minutes. The average sentence length is 12.4 words. The actual distribution of the sentence lengths is shown in Figure 7.1. The word error rate of the first-best transcriptions is 13.27%. The first-best transcription of one of the word lattices is given below:

[silence] Vom Auswärtigen Amt wurde zwar keine ausdrückliche Reise Warnung für Tunesien ausgesprochen <s/> Doch auf der Internet Seite wird zur besonderen Vorsicht gemahnt <s/> Insbesondere bei Menschen Ansammlungen [silence] (breath) <s/> Auch der Betroffene Reiseveranstalter [silence] zieht Konsequenzen <s/>

Besides the tags indicating silence, breath, filler words and potential sentence boundaries, the speech recognition output has certain peculiarities that have to be taken into account by the grammar. In particular, the speech recognizer may split words such as compound nouns (e.g. “Internet Seite” instead of “Internetseite”), acronyms (e.g. “B. M. W.” instead of “BMW”), numbers (e.g. “ein und zwanzig” instead of “einundzwanzig”) and prefix verbs (e.g. “aus handelten” instead of “aushandelten”). This
allows the speech recognizer to produce reasonable transcriptions for compounds that are not covered by the vocabulary.

Split compounds are not counted as errors in the evaluation scheme by [MA03]. The evaluation is based on rewrite rules that normalize both the reference transcriptions and the speech recognition hypotheses. Even though syntactic analysis could help to recover the original word forms, we adopted the original evaluation scheme in our experiments.

7.2 Preprocessing

The segmentation component was trained on the three news shows that have been used for preliminary experiments. The three unseen news shows were automatically segmented into 609 sentence-like units and the best hypotheses were incrementally extracted from each resulting word lattice. As many hypotheses only differ in the tags and in the capitalization of words, this information was removed in a normalization step. The extraction of the best hypotheses was stopped as soon as 100 different normalized hypotheses were found.

The set of all words occurring in the speech recognition hypotheses was computed. This set of words was the basis on which all linguistic resources were developed. No other information about the test data was known to the language engineer in this phase.

For each of the generated word lattices, the reference transcription was determined and annotated with syntactic structure. This information was later used for the cross-validation training of the disambiguation model.

All hypotheses were parsed and the resulting parse forests were stored on disk. For three utterances, some hypotheses could not be parsed exhaustively due to memory limitations\(^1\). These cases were dealt with by removing the first hypothesis that could not be processed completely, as well as all hypotheses that were ranked higher. For one utterance, the processing of the first-best hypothesis failed. In this case, the syntactic information was not considered at all and the first-best hypothesis was chosen as the recognition result.

The part-of-speech tagger was trained on a transformed version of the TIGER corpus [BDH\(^+\)02] as described in Section 5.5.

---

\(^1\)Parsing was aborted as soon as more than 5 000 000 feature structure nodes were created. Thus, the termination criterion was entirely deterministic.

7.3 Basic Setup

In order to make the best use of the test data, we followed a 10-fold cross-validation scheme: the hypothesis reranking model was tested on each fold after being trained on the remaining folds. This procedure was complicated by the fact that training and testing the reranking model is based on the most likely parse tree of each hypothesis. The extraction of the most likely parse trees in turn requires a disambiguation model, which has to be trained by means of an embedded cross-validation cycle.

More formally, let \(f_{\text{test}}\) be the fold to be evaluated and \(S_{\text{training}}\) the set of the remaining folds. The most likely parse trees of the hypotheses in \(f_{\text{test}}\) are extracted by means of a disambiguation model that was trained on \(S_{\text{training}}\). In order to extract the most likely parse trees for a fold \(f \in S_{\text{training}}\), the disambiguation model is trained on the folds \(S_{\text{training}} \setminus \{f\}\). Once this has been done for all folds \(f\) in the training set, the features of the hypothesis reranking model are extracted from the training set and the model parameters are estimated. The hypothesis reranking model can then be applied to the test fold \(f_{\text{test}}\). This procedure guarantees that no information from the test set is used for training, and that the hypothesis reranking model is trained and tested on similar conditions.

The utterances were randomly assigned to the 10 folds. A random assignment is considered to be inappropriate for tasks with highly clustered data, such as part-of-speech tagging. For example, rare words that are related to the topic of a paragraph are likely to occur several times within the same paragraph. It is likely that some of these occurrences are assigned different folds and thus can be learned by the tagger. Consequently, the performance of the tagger is overestimated. We argue that such effects are negligible in our experiments. Our models learn syntactic preferences without considering the actual word forms. The syntactic structure is much more correlated with the current section (anchor speaker, weather report, interview, etc.) than with the topic, and these sections are present in each news show.

The conditional log-linear models were trained with the reranking software presented in [CJ05]. The regularization parameter \(c\) was set to 13 for disambiguation and 30 for hypothesis reranking. These values were found to work well in earlier experiments with similar data and feature sets.
The training data for the disambiguation model was complemented with 447 syntactically annotated broadcast news transcripts from the other three news shows. There is a slight mismatch between this additional training data and the data from our experiments: the transcripts have correct sentence boundaries and represent only the sections with low word error rates (anchor speakers, weather reports and correspondents). Overall, the training data for the disambiguation model amounts to about 990 sentences.

### 7.4 Broadcast News Transcription Results

Table 7.1 shows the main result of this thesis. The word error rate was reduced by 9.7% relative due to automatic segmentation and 100-best reranking with syntactic information. This improvement is statistically significant at a level of less than 0.1% according to both the the Matched Pairs Sentence-Segment Word Error test (MAPSSWE) and the McNemar test on sentence level [GC89].

<table>
<thead>
<tr>
<th>approach</th>
<th>word error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline system</td>
<td>13.27%</td>
</tr>
<tr>
<td>+ syntactic information</td>
<td>11.98% (-9.7% relative)</td>
</tr>
<tr>
<td>100-best oracle</td>
<td>6.32%</td>
</tr>
</tbody>
</table>

Table 7.1: The word error rate after reranking with syntactic information compared to the first-best word error rate of the baseline system and the 100-best oracle error rate.

This result was obtained by combining information from the n-gram language model and syntactic information. In order to estimate the contributions of the two language models in isolation, we performed experiments in which the 100-best reranking involved different sets of features. The baseline was defined as the word error rate that is achieved by always choosing the hypothesis with the highest acoustic score. The contribution of the syntactic information was determined by disabling all features that depend on the n-gram score or the rank of a hypothesis. The performance of the n-gram language model in isolation is given by the word error rate of the baseline speech recognizer. Table 7.2 shows that considering syntactic information in addition to the acoustic score leads to a large improvement that comes close to that for the n-gram language model. This is surprising, considering that the syntactic information is based on syntactic categories rather than actual words. However, word information is not completely missing as n-gram probabilities were involved in selecting the 100 best hypotheses.

It is sometimes suspected that adding a grammar-based component to speech recognition mainly helps in correcting inflectional endings, particularly for highly inflected languages such as German. Inspection of the experimental results suggested that this is not the case. In the following example, the eleventh-best hypothesis was preferred on grounds of syntactic information, and so the words “kritik lehnte” were correctly changed to “für die gelähmte”:

<table>
<thead>
<tr>
<th>ref</th>
<th>es gibt keinen legalen Weg für die gelähmte Britin ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>es gibt keinen legalen weg kritik lehnte britten ...</td>
</tr>
<tr>
<td>11th</td>
<td>es gibt keinen legalen weg für die gelähmte britten ...</td>
</tr>
</tbody>
</table>

Note that in this example, the preferred hypothesis is ungrammatical: the plural noun *briten* does not agree with the preceeding adjective and the determiner. This indicates that syntactic information can be beneficial even if the hypotheses are ungrammatical. Further investigations were carried out to test these assumptions empirically.
For each error introduced or corrected in the reranking step, we checked whether it was due to changing an inflectional ending in the first-best transcription. It was found that only 27% of the net error corrections involved changing inflectional endings. The remaining corrections were due to introducing a stem or a function word that was not present in the first-best hypothesis.

Robustness with respect to out-of-grammar utterances was analyzed by considering all instances for which the reranker preferred an out-of-grammar hypothesis to the first-best hypothesis. It was observed that about 37% of the net error corrections were achieved by choosing a hypothesis with more than one partial parse tree. This confirms that syntactic information also leads to improvements for out-of-grammar utterances.

In order to assess the impact of the number of hypotheses $N$ on the word error rate, we repeated the experiment for all $N$ in $1..100$. We did not try higher values of $N$ as the lexicon is only guaranteed to cover the words of the 100 best hypotheses. The results are shown in Figure 7.2. The 10 best hypotheses alone account for an improvement of about 1% absolute (7.8% relative). For higher values of $N$, the word error rate starts to level off. However, there still seems to be room for further improvement beyond $N=100$.

### 7.5 Influence of Recognition Task

In order to measure the influence of different characteristics of the recognition task, we performed experiments for artificially simplified tasks. The first experiment is identical to the original experiment except for the segmentation. The sentence boundaries were determined manually, and the resulting segments were not split if they were longer than 25 words. This experiment can be regarded as assuming a perfect sentence boundary detection, or as approaching a dictation scenario where sentence boundaries are clearly marked. The goal of the second experiment was to assess the benefit of syntactic information for a lower baseline word error rate. This was achieved by restricting the data to anchor speaker sections, correspondent speech and weather reports, which amounts to a total of 506 utterances. The sentence boundaries were determined automatically.

The results are shown in Table 7.3. Correct segmentation resulted in a relative improvement of 12.2%. The improvement relative to the ex-

![Figure 7.2: The word error rates that result from reranking the $N$ best speech recognition hypotheses, $1 \leq N \leq 100$.](image)

<table>
<thead>
<tr>
<th>approach</th>
<th>word error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>13.27%</td>
</tr>
<tr>
<td>+ syntactic information</td>
<td>11.98% (-9.7% relative)</td>
</tr>
<tr>
<td>+ correct segmentation</td>
<td>11.67% (-12.2% relative)</td>
</tr>
<tr>
<td>baseline (selected sections)</td>
<td>10.91%</td>
</tr>
<tr>
<td>+ syntactic information</td>
<td>9.41% (-13.7% relative)</td>
</tr>
</tbody>
</table>

**Table 7.3:** The relative improvements for two artificially simplified speech recognition tasks. The first experiment assumes correct segmentation and is based on manually determined sentence boundaries. The second experiment is performed on a subset of the data, i.e. the anchor speaker, correspondent or weather report sections.
Experiment with automatic segmentation was only weakly significant: 4.8% according to the MAPSSWE test. However, these results are in line with [MKHO06], who observed that manual segmentation improved the speech recognition performance when using syntactic information from a statistical parser.

Table 7.3 shows that the lower baseline word error rate led to a larger relative improvement. This confirms the intuition that a language model which makes extensive use of non-local information is particularly good at correcting errors within contexts that are largely correct. [Beu07] made a similar observation when varying the out-of-vocabulary rate of the baseline speech recognizer. In his experiments, increasing the out-of-vocabulary rate (and hence increasing the word error rate) almost consistently led to a reduction of the relative improvement.

The influence of the baseline word error rate was further investigated by selecting experimental data on the level of single utterances. A sequence of utterances was created by sorting the 609 utterances by word error rate. For every sub-sequence of 250 consecutive utterances, the baseline word error rate and the relative improvement due to the proposed approach were computed. 10-fold cross-validation was used to train and evaluate the grammar-based language model on the same set of utterances. The results are shown in Figure 7.3. For this artificial experiment, the speech recognition performance degrades rather slowly with increasing baseline word error rate. However, the underlying data sets are unnaturally homogeneous by construction and thus the benefit of syntactic information is overestimated. The poor performance for very low word error rates can be explained by the fact that these data sets contain almost no training examples for which an improvement is actually possible. This results in a large mismatch between the test data and the training data.

A similar experiment was conducted to investigate the effect of varying the oracle word error rate: the 609 utterances were sorted by oracle word error rate and the relative improvement was measured for each sub-sequence of 250 consecutive utterances. Figure 7.4 shows that the benefit of syntactic information quickly degrades with increasing oracle word error rate. In an additional experiment on all 203 utterances with non-zero oracle word error rate, the relative improvement was found to be only about 3%. These results suggest that the proposed approach heavily relies on the availability of the correct speech recognition hypothesis.
It is often objected that a grammar-based language model might not work well for spontaneous speech. In order to get an idea of how much improvement can be expected in such situations, we performed an experiment on the 92 utterances that appeared to be instances of spontaneous speech. Again, 10-fold cross-validation was used to train and evaluate the grammar-based language model on the same set of utterances. It was found that syntactic information reduced the word error rate by 5.6% relative, compared to a baseline of 22.03%. However, these results are by no means conclusive. First, the underlying set of utterances is very small, both for training and for evaluation. Second, the situations with spontaneous speech are often difficult for the baseline speech recognizer because of background noise or insufficient speaker adaptation. The improvements are likely to be larger under more controlled conditions.

### 7.6 Influence of Linguistic Information

In this section, it is investigated how the choice of features and the amount of linguistic information influence the speech recognition performance. The main results are shown in Table 7.4.

Model 1 involved no statistical information on syntactic preferences. The artificial parse trees were extracted by means of the fewest-chunks heuristics, and the hypothesis reranking features were restricted to the recognition score features and the non-prosodic features based on counts of partial parse trees. This corresponds roughly to the information present in [Beu07], except that we do not distinguish between single-word partial parse trees and partial parse trees with two or more words. It can be seen that the lack of statistical information substantially reduces the benefit of our approach. This reduction is statistically significant on a level of <0.1% according to the McNemar test and the MAPSSWE test.

Model 2 uses the disambiguation score and the recognition score features as the only features for hypothesis reranking. This model significantly improves on the model without syntactic preferences (<1.4% according to the MAPSSWE test, <0.1% according to the McNemar test). This model is further improved by adding the features based on partial parse tree counts (model 3) and the features taking prosodic boundaries into account (model 4). If the syntactic categories of the partial parse trees are considered additionally (model 5), the word error rate is already
very close to that of the full model. The difference can be attributed to the syntactic preference features.

In order to assess the benefit of the disambiguation score feature, this feature was removed from the full model. The word error rate of the resulting model (model 6) was significantly lower than that of the full model (<0.3% according to the MAPSSWE test).

We further investigated the importance of the sentence grammar for speech recognition. The original grammar was modified by disabling all rules that participate in the formation of verbal complexes, verb phrases and adjective phrases. The resulting grammar could still derive noun phrases, prepositional phrases and expressions of data and time. In particular, the grammar still represented agreement constraints such as agreement between the determiner, the adjective and the noun, as well as case agreement between a preposition and its complement noun phrase. The full model in combination with this chunk grammar achieved a word error rate of 12.78%, which corresponds to a relative reduction of 3.7%. This reduction is still statistically significant (0.4% according to the MAPSSWE test, 0.1% according to the McNemar test). At the same time, the improvement is significantly lower than that of the full model with the sentence grammar (0.1% according to both the MAPSSWE and the McNemar test).

Finally, we varied the amount of syntactically annotated data for the training of the disambiguation model by means of random subsampling.

### Table 7.4: Word error rates for different sets of features.

<table>
<thead>
<tr>
<th>model</th>
<th>word error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 no statistics</td>
<td>12.70% (-4.3% relative)</td>
</tr>
<tr>
<td>2 score features</td>
<td>12.24% (-7.7% relative)</td>
</tr>
<tr>
<td>3 + non-prosodic partial parses</td>
<td>12.14% (-8.5% relative)</td>
</tr>
<tr>
<td>4 + prosodic partial parses</td>
<td>12.09% (-8.9% relative)</td>
</tr>
<tr>
<td>5 + partial parse types</td>
<td>12.05% (-9.2% relative)</td>
</tr>
<tr>
<td>full model</td>
<td>11.98% (-9.7% relative)</td>
</tr>
<tr>
<td>6 no disambiguation score</td>
<td>12.39% (-6.6% relative)</td>
</tr>
</tbody>
</table>

The influence of this parameter on the word error rate is shown in Figure 7.5. It can be seen that the number of training sentences can be halved without significantly impairing the speech recognition performance. This result confirms our claim that a relatively small amount of syntactically annotated data is sufficient for the training of our model. Note that the hypothesis reranking model also employs syntactic features and thus can compensate for the lack of training data to some degree.

### 7.7 Evaluation of Components

#### 7.7.1 Grammar and Lexicon

In order to assess the coverage of the grammar and the lexicon, the reference transcription of each sentence unit was parsed\(^2\). For each sentence,

\(^2\)As the words occurring in the reference transcriptions are not necessarily covered by the lexicon, missing words were included beforehand.
the parse tree that best matched the reference syntax tree was extracted and examined. A correct complete parse tree was found for 61% of the sentences. Another 7.3% of the sentences were ellipses for which all parts could be correctly analyzed by the parser. Only 3% of the sentences were considered to be clearly ungrammatical in the sense of incorrect language use.

[NKZ+08] evaluated the German DFKI grammar [MK00, Cry03a, Cry05] on newspaper sentences consisting of up to 20 tokens. About 42% of the sentences that were completely covered by the lexicon were found to have at least one complete parse tree. Thus, the actual coverage can be expected to be even lower. These numbers cannot be compared directly as newspaper articles and news show transcripts are different sorts of text. Further, the DFKI grammar and the corresponding lexicon were not developed for parsing newspaper text: according to the figures given by [Cry08], the coverage is currently around 62.5% for a subset of the Verb mobil corpus. However, the results illustrate that the coverage of precision grammars is generally rather low and that the coverage of our grammar is competitive.

It was also attempted to determine the different types of errors that prevented sentences from being analyzed correctly. We manually identified all errors for those sentences for which no correct complete parse tree could be derived. Each error was assigned an error type (e.g. missing multiword lexeme), and the different error types were partitioned into five broad error classes: lexicon errors, grammar problems, missing general constructions I/II and missing idiosyncratic constructions.

Table 7.5 shows the distribution of the errors with respect to the five basic error classes. The lexicon errors account for almost one third of the errors. More than half of the lexicon errors are due to missing verb valence information like "auf etw. abkühlen" and the different error types were partitioned into five broad error classes: lexicon errors, grammar problems, missing general constructions I/II and missing idiosyncratic constructions.

Table 7.5: A rough classification of the errors that occur when parsing the reference transcriptions.

<table>
<thead>
<tr>
<th>error class</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>lexicon errors</td>
<td>30%</td>
</tr>
<tr>
<td>grammar problems</td>
<td>5%</td>
</tr>
<tr>
<td>missing general constructions I</td>
<td>13%</td>
</tr>
<tr>
<td>missing general constructions II</td>
<td>28%</td>
</tr>
<tr>
<td>missing idiosyncratic constructions</td>
<td>24%</td>
</tr>
</tbody>
</table>

Grammar problems occur if the grammar fails to account for a construction that is supposed to be covered by the grammar. For example, the phrase "das depot, das umgeben von wohngebieten ist" cannot be analyzed because the grammar prohibits the interruption of verbal complexes (in this example umgeben ist) by complements or adjuncts.

Lacking coverage of general constructions is captured by two classes of errors. The errors in the first class are due to missing constructions that could be incorporated into a next version of the grammar. These constructions include certain types of appositions, sentential subjects and complements of predicative adjectives, free relative clauses and sentential relative clauses.

The second class of errors is related to general phenomena that are notoriously difficult to formalize or that would give rise to a large amount of ambiguity. These constructions include coordinations (asyndetic coordinations, asymmetric coordinations and coordinations of constituents with different categories), parentheses and ellipses. Another important type of error that falls within this class is the omission of mandatory determiners as auslöser in "auslöser war eine herabstufung des unternehmens".

The last of the basic error classes covers idiosyncratic or rare constructions that are not part of the grammar. Examples are the X für X construction as in the utterance "in hebron durchkämmten die soldaten haus für haus", or the was...wenn construction as in "(…) darüber zu spekulieren, was geschehen würde wenn." The main problem with such constructions is that there are many of them and they each require a ded-
icated grammar rule or a special lexicon entry. In fact, it is not trivial to discover such constructions in the first place and to make a reasonable selection of the constructions to be included into the grammar.

### 7.7.2 Robust Parsing

The benefit of the proposed robust parsing technique was evaluated by performing a comparative experiment with the alternative fewest-chunks heuristics. The training of the disambiguation model was identical for both methods. The alternative experiment differed from the original experiment only in that the disambiguation component chose an artificial parse tree with the minimum number of partial parse trees.

The word error rate achieved with the fewest-chunks heuristics was 12.07%, which corresponds to a relative reduction of 9.0%. Thus, the improvement of the proposed technique over the fewest-chunks heuristics is small and not statistically significant.
Chapter 8

Conclusions

8.1 Discussion

The primary aim of this thesis was to determine whether rule-based linguistic knowledge can improve automatic speech recognition on a broad-domain task. In particular, we were interested in measuring the benefit from precise linguistic constraints as encoded in a precision grammar.

We have proposed a novel approach to integrating a precision grammar into a statistical speech recognition framework. Even though our approach is based on a formal grammar that imposes hard constraints on natural language, this information is applied in a “soft” way: the grammar is complemented with a statistical model representing soft constraints, and hard constraints can be violated. The approach was implemented and applied to a German broadcast news transcription task. Our experiments confirmed that the use of a precision grammar can significantly improve large-vocabulary speech recognition, even for a broad domain and a competitive baseline system.

The automatic transcription of broadcast news is complicated by out-of-vocabulary words, missing sentence boundaries and (at least to some extent) bad acoustic conditions. Thus, the chosen task is by no means simple. Nevertheless, syntactic information may be particularly beneficial for broadcast news transcription as this task involves mostly prepared grammatical speech. We do not provide a conclusive evaluation on spontaneous speech, which is the main focus of speech recognition industry. We believe that our approach would still yield some improvement on such tasks, though it would certainly be lower than the gains reported in this thesis. Although many spontaneous utterances may be ungrammatical, they are likely to contain syntactic units to which hard linguistic constraints can be applied, e.g., noun phrases, prepositional phrases and subordinate clauses. It might also be possible to adapt to the nature of spontaneous speech by extending the grammar or by using preprocessing components that deal with repair phenomena or interjections. The latter technique was applied in [MAD+95].

We have also investigated different properties of our approach. In particular, it was observed that grammaticality information alone led to a relatively small (though still statistically significant) improvement. However, the absolute improvement was more than doubled by adding statistical information, even though the statistical models were trained with a rather small amount of data (about 450 sentences/5200 words of syntactically annotated text and 550 N-best lists). This effect may be explained by the precision of the grammar and the lexicon: the grammar still accepts many implausible sentences, but in order to do so, it has to resort to a number of specific constructions that can easily be penalized by the statistical models. Examples of such constructions are prenominal genitives and substantivized adjective phrases, which indeed are strongly penalized by the log-linear models. These results suggest that the qualitative information expressed in the grammar and the lexicon greatly reduces the need for quantitative information, i.e. treebank data.

The small amount of treebank data is a distinctive characteristic of the proposed approach in comparison to the related work of [CRS05] and [MKHO06] (see Section 4.6). These approaches are based on statistical parsers that were trained on about one million words of syntactically annotated text. It is difficult to compare the effort of creating a treebank to that of writing a precision grammar and acquiring a large lexicon. Both tasks are very laborious and require expert knowledge. It is possible that qualitative information can be transferred to new domains more easily, but this is a rather speculative claim.

Another characteristic is that our approach does not consider head word information at all. Thus, semantic aspects like common combinations of a verb and the head word of its subject are not captured. This is again in contrast to the work of [CRS05] and [MKHO06], who made
extensive use of this sort of information. We conclude from this that the proposed approach is, to some degree, orthogonal to statistical methods that consider head-to-head dependencies. How large the actual gain from combining such statistical methods with rule-based approaches would be and how head word information could be incorporated into our approach are open questions.

8.2 Outlook

In this thesis, it was attempted to investigate the benefit of linguistic information. The proposed approach is based on an N-best reranking scheme as this architecture imposes only minimal constraints on the linguistic components. In the context of this work, the primary objective was the precision of the linguistic constraints rather than industrial applicability. It was aimed at obtaining an estimate of how much improvement this kind of information can yield, though the presented results cannot be regarded as an upper bound.

Yet, an interesting question is how the proposed approach could be applied to a real speech recognition system. This problem is twofold. First, the parsing of the N-best hypotheses (or the word lattices) would have to be much more efficient. This might be achieved by approximating a precision grammar along the lines of [KK00]. Alternatively, it could be feasible to use a less restrictive grammar and move some of the expensive linguistic constraints (e.g. verb valence) to the disambiguation model and the hypotheses reranking model.

The second problem is the acquisition of lexical resources. The manual creation of a precise lexicon was viable for the small-scale experiment presented in this thesis. However, this would most probably not be the case for a speech recognizer vocabulary with hundreds of thousands of word forms. Here, one option might be to specify only those syntactic properties that can reliably be extracted from text corpora and to describe the remaining properties by means of default values or underspecification. Preferences for these properties could be modeled by the statistical models, falling back to less reliable features extracted from text corpora.

Apart from efficiency and scalability, there is another reason why relaxing the hard linguistic constraints appears to be worthwhile. Even though this thesis has placed a strong emphasis on precise linguistic knowledge,
Appendix A

Grammar Coverage

In Section 5.4, the German grammar was described in terms of informally stated grammar rules. However, the precision and the coverage of the grammar also depend on the specifics of the grammar rules, on the type hierarchy and on the actual lexicon entries. Also, the domain-specific sub-grammars were omitted from the discussion. In order to provide a more detailed outline of the grammar’s coverage, this appendix presents an illustrative set of accepted and rejected word sequences. The word sequences that are preceded by an asterisk symbol (*) are correctly predicted to be ungrammatical, whereas all other word sequences are correctly predicted to be grammatical. The word sequences are based on a set of about 1200 test sentences for checking grammar consistency. The complete set is available at http://www.tik.ee.ethz.ch/~spr/hpsg0409/.

A.1 Verbal Projections

Main Clauses

ger gab ihr das Buch – complement in the prefied
*gestern gab er ihr das Buch – modifier in the prefied
*gestern er gab ihr das Buch – multiple constituents in the prefied
*der Mann jedoch schlief – adverb connector
*der Mann bald schlief – regular adverbs cannot occupy this position

gib mir das Buch – imperative clause
*mir gib das Buch – imperative clauses require an empty prefied

Subordinate Clauses

dass es regnet – dass clause
weil es regnet – adverbial clause
ob es regnet – interrogative clause
von welchem Mann er spricht – interrogative clause
*der Mann den ich kenne – relative clause
*der Mann ich den kenne – relative clauses must start with relative phrase
*die Frau mit deren Kindes Puppe er sprach – embedded relative pronoun
*die Frau mit dessen Kindes Puppe er sprach – agreement violation
*die Puppe das Kind mit welcher er kannte – invalid embedding
*der Mann den zu kennen er nicht zugeben wollte – pied piping

Verbal Complex

er hat ihr das Buch zu geben versucht – left and right sentence bracket
*dass er ihr das Buch zu geben versucht hat – without left sentence bracket
dass er ihr das Buch haben wollen – auxiliary flip (Oberfeldumstellung)
zu schlafen versuchen wird er wollen – fronted partial verb phrase
*dass er versucht den Mann zu verstehen – extraposition
dass den Mann zu verstehen er versucht – intraposition
dass er den Mann zu verstehen versucht – coherent construction
dass den Mann er zu verstehen versucht
*den Mann hat er versucht zu verstehen – extraction from extraposited VP
er fiel ihr auf – verb prefix in the right sentence bracket

Raising and Control

dass es zu regnen schien – subject raising
*dass er zu regnen schien
dass er es regnen sah – object raising (accusatvus cum infinitivo)
*dass er ihn regnen sah
dass ich versuchte mich zu beeilen – subject control
*dass ich versuchte sich zu beeilen
dass er mir empfahl mich zu beeilen – object control
*dass er mir empfahl sich zu beeilen
A.1 Verbal Projections

dass er verstanden worden ist – werden passive

dass er zu verstehen versucht worden ist – remote passive

dass jetzt geschlafen wird – impersonal passive

dass der Mann zu kennen ist – modal infinitive

dass ich ihr ein Buch geben ließ – lassen passive

dass er ein Haus vermacht bekam – dative passive

*dass der Kuchen gegessen bekam

Copula

sie ist schön – predication
diese Hunde sind eine Plage – identity

seiner Frau ist er treu – predicative adjective with complement

*seiner Frau ist er stolz

dass er schon – predicative adjective with complement

dass er schon – predicative adjective with complement

Complements

dass er an einen Zufall glaubt – prepositional phrase complement

*dass er gegen einen Zufall glaubt

dass er behauptete es regne – main clause complement

dass er fragte ob es regnet – interrogative clause complement

dass er weiß dass es regnet – dass clause complement

dass er es weiß dass es regnet – substituting expletive pronoun

dass er darunter leidet dass es regnet – substituting pronominal adverb

*dass er darüber leidet dass es regnet

dass er den Wein in die Tonne schützt – directional complement

dass er den Wein dorthin schützt – directional complement

gestern standen sie Schlange – lexicalized accusative object

*gestern standen sie die Schlange

*gestern sassen sie Schlange

dass er es in Kauf nimmt – lexicalized prepositional object

*dass er es in notwendigen Kauf nimmt

*dass er es in Kauf gibt

A.2 Noun Phrases

Nouns

der Mann – regular noun
der Beamte – noun with adjectival inflection
ein Beamter

*ein beamter Mann
das Gute – nominalized adjective
das vermeintlich Gute – nominalized adjective

*das Prima – nominalized adjectives must be inflected
das Essen des Apfels – nominalized infinitive

Complements

die Tatsache dass es regnet – dass clause complement
die Behauptung es hätte geregnet – main clause complement
die Frage ob es regnet – interrogative complement

*die Frage dass es regnet

Adjuncts

das kleine Kind – attributive adjective
der Mann mit dem Hund – attributive prepositional phrase
der Mann den ich kenne – relative clause
der Mann selbst – stressed focus particle
der Mann hier – locational adverb
der Weg dorthin – directional adverb
der Bedenknträger Gerhard Schröder – close apposition

Attributive Genitives

der Hund des Mannes – postnominal genitive
des Pudels Kern – prenominal genitive
die Produktion weissen Mehls – sufficient case marking

*die Produktion weissen – insufficient case marking
die Produktion weisser Schokolade
die Produktion weisser

der Traum manchen Mannes
A.3 Adjective Phrases

Expressions of Quantity

mehrere Tonnen Zucker – expression of quantity with mass noun
eine Tonne Glühbirnen – expression of quantity with plural noun
*eine Tonne die Glühbirnen – determiner omission is mandatory
*eine Tonne Glühbirne – determiner omission must be possible
*eine Tonne von mir Zucker – measure must be adjacent to substance

Temporal Adverbials

er schlief einige Stunden – noun phrase as a temporal adverbial
*eher schließ einigen Stunden – adverbial use requires accusative case

Complements

der ihr dankbare Mann – dative noun phrase
das wäre einen Versuch wert – accusative noun phrase
er ist seines Erfolges gewiss – genitive noun phrase
der auf sie stolze Mann – prepositional phrase
ein weit über zwei Tonnen schwerer Apfel – graduative complement
der in diesem Haus wohnhafte Mann – locational complement
*einer wohnhafte Mann

Adjectival Participles

das sinkende Schiff – present participle
der schlafende Mann – present participle
das gesunkene Schiff – past participle
*der geschlafene Mann – not possible for intransitive, unergative verbs
das ihr von uns überreichte Geschenk – participle with complements
das ihr zu überreichende Geschenk – modal infinitive
der ihr das Geschenk überreichen wollende Mann – verbal complex

A.4 Prepositional Phrases

Structure

auf dem Berg – preposition
ihm zufolge – postposition
von jener Stunde an – circumposition
bis auf diesen Berg – complex adposition
bis zu diesem Berg hin – complex adposition

Complements

ich habe ihn für dankbarer gehalten – predicative adjective
es regnete von überall her – locational adverb
es regnete nach oben – locational adverb
er fuhr nach Berlin – geographical name

Modifiers

weit über den Berg warf er die Tonne – modified directional preposition
*bald über den Berg warf er die Tonne – only possible with specific adverbs
*weit über das Wetter freute er sich – only possible for directional PPs
hier in diesem Haus lebte er – modified locational preposition
*hier in diesem Vorwurf gipfelte seine Rede
wenige Meter neben dem Baum schläft ein Hund – expression of quantity

Lexicalization

aus diesem Grund schläft er – adverbial use of aus ... Grund
aus den gegebenen Gründen schläft er – syntactic variability
*aus diesem Rat schläft er
A.5 Coordination

Structure

der Mann und die Frau – simple coordination
sowohl der Mann als auch die Frau – correlative coordination
er freute sich auf Frau und Kinder – determiner omission in coordination

Conjuncts

er schläft denn er ist müde – sentences
er wird entweder den Apfel essen oder schlafen – saturated verb phrases
er wird den Apfel entweder essen oder ihr geben – unsaturated verb phrases
er will und kann den Apfel essen – modal verbs
der Mann dessen Frau und dessen Kinder er kennt – relative phrases
keiner wusste ob und wann er ankommt – interrogative phrases
*ihr ist er lieb und teuer – predicative adjectives
*ihr ist er lieb und nett
er schläft auf oder unter dem Dach – prepositions
es regnete durch das Dach und in das Haus – directional PPs

A.6 Extraposition

Extraposed Material

er hat die Frage erwartet ob es regnet – complement clause of noun
er hat es gewusst dass es regnet – complement clause of verb
er hat es gewusst dass es regnet – substituted complement clause
er hat darauf gewartet dass es regnet – substituted complement clause
er hat lange gewartet auf diesen Moment – prepositional object
er hat einen Hund gesehen der schläft – relative clause

Extraposition Sites

er hat das Futter des Hundes gegessen der schläft – deep embedding
Hunde streicheln die schlafen will er nicht – extraposition within prefield
dass des Hundes wegen der schläft sie schrie – adjacent to postposition
*i dass des Hundes wegen sie der schläft schrie – invalid extraposition site

A.7 Subgrammars

Numerals

neunzehnhundertneunzig – spelled cardinal numbers
neunzehnhundert ein und neunzig – splitting of spelled cardinal numbers
aus tausendundeiner Nacht – case agreement
mit zwei Millionen einhunderttausend Euro – millions
mit hundertundeiner Million Euro – case and number agreement
hundert kleine Kinder – indeterminate number
hunderte kleine Kinder – indeterminate number
hunderte von kleinen Kindern – indeterminate number
ein bis zu mehreren hundert Tonn – numeral expression
drei bis vier Tonn – numeral expression
minus zwei komma acht eins – decimal fraction
sein zweiausachtzigter Geburtstag – ordinal number
ein zweiausachtzigjähriger Mann – adjective with numeral part

Expressions of Date

nächsten Montag wird es regnen
Montag den ersten Juli wird es regnen
am Montag dem ersten Juli zweitausendzehn wird es regnen
morgen Nachmittag
gegen Abend
seit Mitte Januar vergangenen Jahres
am Dienstag kommender Woche
Expressions of Time

\begin{itemize}
  \item um ein Uhr null fünf
  \item von halb eins bis ein Viertel nach drei Uhr
  \item seit drei Viertel acht
\end{itemize}

Personal Names

\begin{itemize}
  \item der Rat Gerhard Schröders – inflection of personal names
  \item der Rat des Herrn Gerhard Schröder – inflected title
  \item der Rat des Doktor Gerhard Schröder – uninflected title
  \item der Rat Doktor Gerhard Schröders
  \item *der Rat des Doktor Gerhard Schröders
\end{itemize}
Bibliography


Curriculum Vitae

1978  Born in Winterthur, Switzerland

1985-1990  Primarschule in Mülligen AG

1990-1994  Bezirksschule in Windisch

1994-1998  Kantonsschule in Aarau

1998-2003  Studies in computer science at ETH Zürich

2003  Diploma in computer science (dipl. ing.)

2003-2009  Research assistant and PhD student at the Speech Processing Group, Computer Engineering and Networks Laboratory, ETH Zürich