Tagging and Parsing with Cascaded Markov Models

Automation of Corpus Annotation

Thorsten Brants

April 1999

Dissertation zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Technischen Fakultät der Universität des Saarlandes.
Betreuer: Prof. Dr. Hans Uszkoreit

Thorsten Brants
Universität des Saarlandes
FR 8.7 Computerlinguistik
Postfach 17 11 50
D-66041 Saarbrücken
thorsten@coli.uni-sb.de

Summary

This thesis presents new techniques for parsing natural language. They are based on Markov Models, which are commonly used in part-of-speech tagging for sequential processing on the word level. We show that Markov Models can be successfully applied to other levels of syntactic processing. First, two classification tasks are handled: the assignment of grammatical functions and the labeling of non-terminal nodes. Then, Markov Models are used to recognize hierarchical syntactic structures. Each layer of a structure is represented by a separate Markov Model. The output of a lower layer is passed as input to a higher layer, hence the name: Cascaded Markov Models. Instead of simple symbols, the states emit partial context-free structures. The new techniques are applied to corpus annotation and partial parsing and are evaluated using corpora of different languages and domains.

Kurz-Zusammenfassung

Ausgehend von Markov-Modellen, die für das Part-of-Speech-Tagging eingesetzt werden, stellt diese Arbeit Verfahren vor, die Markov-Modelle auch auf weiteren Ebenen der syntaktischen Verarbeitung erfolgreich nutzen. Dies betrifft zum einen Klassifikationen wie die Zuweisung grammatischer Funktionen und die Bestimmung von Kategorien nichtterminaler Knoten, zum anderen die Zuweisung hierarchischer, syntaktischer Strukturen durch Markov-Modelle. Letzteres geschieht durch die Repräsentation jeder Ebene einer syntaktischen Struktur durch ein eigenes Markov-Modell, was den Namen des Verfahrens prägt: Kaskadierte Markov-Modelle. Deren Zustände geben anstelle atomarer Symbole partielle kontextfreie Strukturen aus. Diese Verfahren kommen in der Korpusannotation und dem partiellen Parsing zum Einsatz und werden anhand mehrerer Korpora evaluiert.

Erklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Ort, Datum (Unterschrift)

Acknowledgements

I would like to thank the large number of people who were involved, directly or indirectly, in creating this thesis. First of all, my thanks go to my partner and friend, Sabine Kramp. She supported the work on this thesis in many ways, read several versions, was encouraging and tolerant, and made sure that I occasionally left the computer during the last period of work.

I would like to thank my advisor Hans Uszkoreit for providing an inspiring and well-equipped environment at the department of Computational Linguistics in Saarbrücken, from which this thesis has highly benefitted. Furthermore, he devoted a lot of time to intensive talks about and comments on earlier versions.

I am grateful to Wolfgang Wahlster for being my second reviewer. I would also like to thank Yves Schabes who read earlier versions of this thesis and gave valuable comments. Furthermore, I acknowledge the support of Johannes Engelkamp when work on this thesis began.

Many thanks go to Wojciech Skut for long discussions on this work, and for detailed comments on earlier versions of this thesis. Matthew Crocker also provided valuable comments, for which I would like to thank him.

I would also like to thank Oliver Plaehn, who implemented *Annotate*, the graphical annotation tool. This tool enabled the application of the methods presented in this thesis and facilitated the efficient annotation of large amounts of corpus data.

I am very grateful to the annotators of the NEGRA project: Roland Hendriks, Meike van Hoorn, Kerstin Klöckner, Sabine Kramp, Cordula Preis and Bernd-Paul Simon. They not only generated valuable corpus resources, but also intensively tested the presented methods.

I am indepted to the other projects that have decided to annotate their data with the help of these methods and to all users of my statistical part-of-speech tagger TnT.

This thesis benefitted from many other people, in particular Tania Avgusitnova, Rens Bod, Stephan Busemann, Bertold Crysmann, Denys Duchier, Jason Eisner, Gregor Erbach, Erhard Hinrichs, Lars Konieczny, Brigitte Krenn, Daniela Kurz, Robert MacIntyre, Anne-Marie Mineur, Martin Müller, Klaus Netter, Günther Neumann, Karel Oliva, Christer Samuelsson, Gert Smolka, Rosmary Stegmann, Petra Steiner, Holger Stenzhorn, Ann Taylor, Theo Vosse, and Mats Wirén.

Thanks go to the pioneers of the Penn Treebank for their efforts in building the treebank and for making the data available, and to the Verbmobil project for providing annotated German and English Corpora.

This research was carried out in the Graduiertenkolleg Kognitionswissenschaft Saarbrücken, and in the project NEGRA of the Sonderforschungsbereich 378. For this, I acknowledge the support of the Deutsche Forschungsgemeinschaft.

Last but not least I would like to thank my family and friends for continuous support and encouragement during the work on this thesis.

Abstract

The methods presented in this thesis aim at automation of corpus annotation and processing of large corpora. Automation enables efficient generation of linguistically interpreted corpora, which on the one hand are a pre-requisite for theoretical linguistic investigations and the development of grammatical processing models. On the other hand, they are the basis for further development of corpus-based taggers and parsers and thereby take part in a bootstrapping process.

The presented methods are based on Markov Models, which model spoken or written utterances as probabilistic sequences. For written language processing, part-of-speech tagging is probably their most prominent application, i.e., the assignment of morpho-syntactic categories to words. We show that the technique used for part-of-speech tagging can be shifted to higher levels of linguistic annotations. Markov Models are suitable for a broader class of labeling tasks and for the generation of hierarchical structures.

While part-of-speech tagging assigns a category to each word, the presented method of tagging grammatical functions assigns a function to each word/tag pair. Going up in the hierarchy, Markov Models determine phrase categories for a given structural element.

The technique is further extended to implement a shallow parsing model. Instead of a single word or a single symbol, each state of the proposed Markov Models emits context-free partial parse trees. Each layer of the resulting structure is represented by its own Markov Model, hence the name *Cascaded Markov Models*. The output of each layer of the cascades is a probability distribution over possible bracketings and labelings for that layer. This output forms a lattice and is passed as input to the next layer.

After presenting the methods, we investigate two applications of Cascaded Markov Models: creation of resources in corpus annotation and partial parsing as pre-processing for other applications.

During corpus annotation, an instance of the model and a human annotator interact. Cascaded Markov Models create the syntactic structure of a sentence layer by layer, so that the human annotator can follow and correct the automatic output if necessary. The result is very efficient corpus annotation. Additionally, we exploit a feature that is particular to probabilistic models. The existence of alternative assignments and their probabilities are important information about the reliability of automatic annotations. Unreliable assignments can be identified automatically and may trigger additional actions in order to achieve high accuracies.

The second application uses Cascaded Markov Models without human supervision. The possibly ambiguous output of a lower layer is directly passed to the next layer. This type of processing is well suited for partial parsing (chunking), e.g., the recognition of noun phrases, prepositional phrases, and their constituents. Partial parsing delivers less information than deep parsing, but with much higher accuracy and speed. Both are important features for processing large corpora and for the use in applications like message extraction and information retrieval.

We evaluate the proposed methods using German and English corpora, representing the domains of newspaper texts and transliterated spoken dialogues. In addition to standard measures like accuracy, precision, and recall, we present learning curves by using different amounts of training data, and take into account selected alternative assignments. For the tasks of part-of-speech tagging and chunking German and English corpora, our results (96.3% – 97.7% for tagging, 85% – 91% recall, 88% – 94% precsision for chunking) are on a par with state-of-the-art results found in the literature. For the tasks of assigning grammatical functions and phrase labels and the interactive annotation task, our results are the first published.

The presented methods enabled the efficient annotation of the NEGRA corpus as their first practical application. Now, they are being successfully used for the annotation of several other corpora in different languages and domains, using different annotation schemes.

Zusammenfassung

Die vorliegende Arbeit ist mit der Zielsetzung der Automation von Korpusannotation sowie der Verarbeitung großer Textkorpora entstanden. Erst durch Automation ist ein effizienter Aufbau linguistisch interpretierter Korpora möglich. Diese Korpora sind zum einen eine wichtige Voraussetzung für theoretische linguistische Untersuchungen und den Aufbau von grammatischen Verarbeitungsmodellen. Zum anderen sind sie in einem Bootstrapping-Prozeß wiederum die Basis für die Weiterentwicklung korpusbasierter Tagger und Parser.

Die vorgestellten Verfahren basieren auf Markov-Modellen. Sie modellieren natürlichsprachliche Äußerungen als probabilistische Signalfolgen. Für die Verarbeitung geschriebener Sprache ist die wohl bekannteste Anwendung die Zuordnung morphosyntaktischer Kategorien zu Wörtern, das Part-of-Speech-Tagging. Die vorliegende Arbeit zeigt, daß die gleiche Technik, die beim Part-of-Speech-Tagging verwendet wird, auch auf weiteren Ebenen der syntaktischen Verarbeitung Anwendung finden kann. Mit Markov Modellen können sowohl eine größere Gruppe von Klassifikationsproblemen behandelt werden als auch hierarchische Strukturen erzeugt werden.

So werden mit Markov-Modellen den Wörtern und syntaktischen Kategorien grammatische Funktionen zugeordnet. Eine weitere Ebene höher werden den Wurzeln von gegebenen Teilbäumen phrasale Kategorien zugeordnet. Das Verfahren wird weiter ausgebaut, um auch strukturelle Elemente zu erkennen und ergibt schließlich, basierend auf Markov-Modellen, ein neues Modell für die flache syntaktische Verarbeitung. Jeder Zustand gibt anstelle einzelner Wörter oder einzelner Symbole partielle kontextfreie Bäume aus. Jede Ebene der berechneten Struktur wird durch ein eigenes Markov-Modell repräsentiert, woraus sich der Name *Kaskadierte Markov-Modelle* ableitet. Die Ausgabe jeder Ebene der Kaskaden ist eine Wahrscheinlichkeitsverteilung über strukturelle Elemente und deren Kategorien. Diese Ausgabe bildet einen Verband und wird als Eingabe zur nächsthöheren Ebene gereicht.

Nach der Vorstellung der Methoden werden zwei Anwendungen Kaskadierter Markov-Modelle untersucht: die Erstellung von Ressourcen in der Korpusannotation sowie partielles Parsing als Vorverarbeitung für andere Anwendungen.

Im hier vorgestellten neuen Ansatz für die Korpusannotation interagieren ein menschlicher Annotierer und Kaskadierte Markov-Modelle und erzeugen Ebene für Ebene syntaktische Strukturen. Der Annotierer folgt so dem Aufbau und greift gegebenenfalls korrigierend ein, was eine sehr effiziente Annotation erlaubt. Die Existenz alternativer Annotationen sowie deren Wahrscheinlichkeiten geben zusätzlich Aufschluß über die Verläßlichkeit einer bestimmten Zuordnung. Unzuverlässige Zuordnungen können so automatisch erkannt werden und zusätzliche Aktionen zu deren Behandlung auslösen.

In der zweiten Anwendung laufen Kaskadierte Markov-Modelle ohne menschliche Überwachung. Die ambige Ausgabe einer niedrigeren Ebene wird als Eingabe an die nächsthöhere Ebene gereicht. Diese Art der Verarbeitung ist sehr gut geeignet für partielles Parsing (Chunking). Es umfaßt zum Beispiel die Erkennung von Nominal- und Präpositionalphrasen sowie deren Teilkonstituenten. Das liefert zwar weniger Information als tiefes Parsing, dafür aber mit weitaus größerer Genauigkeit und Geschwindigkeit. Beides ist eine wichtige Voraussetzung für die Verarbeitung großer Korpora und dem Einsatz in Anwendungen der Sprachtechnologie.

Die vorgestellten Methoden werden anhand von vier Korpora evaluiert. Wir verwenden jeweils ein deutsches und ein englisches Korpus für geschriebene und transkribierte gesprochene Sprache. Zusätzlich zu den Basis-Vergleichswerten "Accuracy", "Precision" und "Recall" werden Lernkurven sowie die Zuweisung ausgewählter alternativer Kategorien betrachtet. Für das Part-of-Speech-Tagging werden 96.3% – 97.7% Korrektheit erreicht, für das Chunking 84% – 90% Recall und 88% – 94% Precision, was dem aktuellen Stand für andere Techniken entspricht. Für die Zuweisung grammatischer Funktionen und Phrasenkategorien sowie für die interaktive Annotation sind die hier präsentierten Ergebnisse die ersten ihrer Art.

Darüberhinaus finden die Methoden erfolgreich praktischen Einsatz beim Aufbau des NEGRA-Korpus, zu dessen Entstehung sie maßgeblich beigetragen haben. Auch für die Annotation mehrerer weiterer Korpora werden sie eingesetzt und sorgen so für den effizienten Aufbau von Ressourcen.

Contents

Summary, Kurz-Zusammenfassung **i**

Erklärung **iii**

Acknowledgements **v**

Abstract **vii**

Zusammenfassung **ix**

1 Introduction **7**

1.1 Motivation 8

1.1.1 Corpus Annotation 9

1.1.2 Partial Parsing and Chunking 11

1.2 Contribution of this Thesis 12

1.3 Outline 14

2 Definitions **15**

2.1 Frequently Used Notations 15

2.2 Markov Models 16

2.2.1 First Order Markov Models 16

2.2.2 Higher Order Markov Models 17

2.2.3 Dynamic Programming 18

2.2.4 Parameter Generation 20

2.3 Stochastic Context-Free Grammars 21

2.3.1 Chart Parsing 22
2.3.2 Parameter Generation 24

3 Related Work 25
3.1 Part-of-Speech Tagging 25
3.1.1 *n*-gram Models 28
3.1.2 Estimating Parameters 28
3.1.3 The sparse data problem 29
3.1.4 Handling of Unknown Words 31
3.1.5 Implementations 31
3.2 Assignment of Grammatical Functions 32
3.3 Stochastic Natural Language Parsing 33
3.3.1 Context-Free Parsing 33
3.3.2 Propagating Lexical Information 36
3.4 Partial Parsing 38
3.4.1 Tagging with Structural Tags 38
3.4.2 Finite State Cascades 39
3.5 Markov Models and Weighted Finite-State Transducers 41
3.6 Automation of Corpus Annotation 42

4 Tagging and Parsing with Markov Models 45
4.1 Introduction 45
4.2 Part-of-Speech Tagging 47
4.3 Alternatives and Reliability 49
4.3.1 Reliability 50
4.3.2 Remaining ambiguity 52
4.4 Tagging Grammatical Functions 53
4.4.1 The Method 54
4.4.2 Encoding of States in the Markov Model 57
4.4.3 Morphological Information 58
4.5 Assigning Phrase Categories 60
4.5.1 The Method 60

4.5.2 Encoding of States 62
4.6 Cascaded Markov Models 63
4.6.1 Tagging Lattices 63
4.6.2 The Method 67
4.6.3 Selecting the Best Phrase Hypothesis 71
4.6.4 Parameter Generation 73

5 Applications of Cascaded Markov Models 75
5.1 Interactive Corpus Annotation 75
5.1.1 Interleaved Automatic and Manual Annotation 75
5.1.2 Selecting the Best Phrase Hypothesis 79
5.1.3 Label Selection 82
5.1.4 Graphical Annotation Tool 83
5.2 Partial Parsing with Cascaded Markov Models 85
5.2.1 The Layered Partial Parsing Model 85
5.2.2 A Processing Example 88
5.2.3 Finding Top-Level Chunks 91

6 Evaluation Methodology 93
6.1 Rules of Evaluation 94
6.2 Tagging Accuracy 96
6.3 Crossing Brackets, Recall and Precision 97
6.4 Exact Match 98
6.5 n-best and Alternative Assignments 98

7 Evaluation Corpora and Results 101
7.1 Corpora 102
7.1.1 NEGRA Corpus 102
7.1.2 Penn Treebank 103
7.1.3 Verbmobil Corpus, German and English Parts 103
7.2 Part-of-Speech Tagging 103
7.2.1 Part-of-Speech Tagging Accuracy 105

7.2.2 Learning Curves for Part-of-Speech Tagging 106
7.2.3 Remaining Ambiguity for Part-of-Speech Tagging 110
7.2.4 Most Frequent Tagging Errors 112
7.2.5 Summary of Part-of-Speech Tagging Results 115
7.3 Tagging Grammatical Functions 116
7.3.1 Accuracy of Assigning Grammatical Functions 116
7.3.2 Learning Curves for Grammatical Functions 117
7.3.3 Remaining Ambiguity for Grammatical Functions 121
7.3.4 Most Frequent Assignment Errors 122
7.3.5 Summary of Results for Assigning Grammatical Functions . . 125
7.4 Assigning Phrase Categories 126
7.4.1 Accuracy of Assigning Phrase Categories 126
7.4.2 Learning Curves for Phrase Categories 126
7.4.3 Remaining Ambiguity for Phrase Categories 128
7.4.4 Most Frequent Assignment Errors 131
7.4.5 Summary of Results for Assigning Phrase Categories 135
7.5 Cascaded Markov Models – Interactive Mode 136
7.5.1 Node Results 136
7.5.2 Learning Curves 139
7.5.3 Summary of Interactive Results 140
7.6 Partial Parsing with Cascaded Markov Models 143
7.6.1 Partial Parsing Results 143
7.6.2 Learning Curves 144
7.6.3 Summary of Partial Parsing Results 147

8 Conclusions **151**
8.1 Contributions 151
8.2 Future Directions 152

References **155**

Appendix **167**

A Tagsets **167**

A.1 Stuttgart-Tübingen-Tagset (Parts-of-Speech) 167

A.2 NEGRA Corpus 169

A.2.1 NEGRA Corpus – Phrase Categories 169

A.2.2 NEGRA Corpus – Grammatical Functions 170

A.3 Penn Treebank 172

A.3.1 Penn Treebank – Part-of-Speech Tags 172

A.3.2 Penn Treebank – Phrase Categories 173

A.3.3 Penn Treebank – Function Tags 174

Chapter 1

Introduction

Statistical and corpus-based methods are currently very successful in speech and language processing. These methods learn from information that is either explicitly or implicitly contained in large corpora. The resulting models are robust in the sense that they cope with unknown words and ill-formed input, and they are efficient since there are efficient algorithms to process them.

Robustness and efficiency are important characteristics of language *performance* models. The domain of a performance model is the production and reception of sentences. These models are opposed to (or complemented by) language *competence* models which aim at characterizing a set of well-formed sentences in a compact and non-redundant way. In one sentence: competence models examine what *could* be said, performance models examine what *actually is* said.

In addition to robustness and efficiency, performance models are also concerned with limitations that are found in human language processing. For instance, the well-known fact that center self-embedded clauses which have a depth of three, which is not very much, or more are difficult to understand is often represented in performance models. Statistical models introduce another property which distinguishes performance and competence models. The concept of grammaticality is no longer a binary one, but a rating on a continuous scale, i.e., sentences and analyses of sentences are ranked "better" or "worse" compared to other sentences or analyses. Statistical models decide on the ranking of a sentence based on frequency. The more frequent a phenomenon, the higher it is ranked. This notion of frequency is incorporated into an increasing number of theories of human language processing.

1.1 Motivation

Corpora collect sentences that have actually been produced in (more or less) natural and domain-specific situations, either by writing or by speaking. They reflect the type of sentences and phenomena which are produced and the frequency thereof. At the same time they represent sentences and phenomena that are perceived by groups of people. For some corpora, e.g., newspaper texts, the group of recipients is much larger than the group of producers. All this makes a corpus a valuable empirical basis for investigations on human language performance, and the information contained in corpora can contribute to language performance models.

Parsing natural language in general is very difficult, but parsing a specific sentence is relatively easy for a human being, although one still needs to handle the problem that different individuals may have different opinions on the exact interpretation of a specific sentence. Manual or semi-automatic parsing of sentences serves two purposes. On the one hand, we gain insight into our language by linguistically analyzing sentences that were produced under natural conditions. On the other hand, while there exist methods for training on raw corpora, usually the best results for parsing are obtained by using linguistically interpreted corpora, so that the system can learn from examples and their interpretations.

Markov Models are a specific class of probabilistic models that learn from corpora. The process of learning is often referred to as *training*. They were first introduced by Andrei A. Markov for a corpus-linguistic purpose[1]: modeling transition probabilities of letter sequences in Russian literature (Markov, 1913). Letters were modeled as random events that depend on a small number of immediately preceding letters. Nowadays, the same idea is successfully used at several levels of speech and language processing.

In speech recognition, phonemes are recognized by exploiting transitional probabilities of acoustic features. Words are recognized by using transitional probabilities of phonemes. Sentences are recognized by using transitional probabilities of words.

[1]Although the term *corpus linguistics* was not coined at that time.

In language processing, syntactic categories of words are recognized by observing their transitional probabilities in sequences. The common feature of these techniques is that they are applied to sequences of signals, and each signal is modeled to be dependent on a finite (and usually very short) history.

The big advantage of Markov Models is that they are fast. Their time complexity is linear to the length of the input. Furthermore, they have been shown to yield very accurate results.

Therefore, our intention is to extend the techniques of language modeling with Markov Models so that they can be applied above the word level. Using these extensions, we recognize different types of labels in addition to parts-of-speech, and we use cascades of Markov Models to recognize syntactic structures.

Research in this area of syntactic processing is motivated by two types of applications. The first one is semi-automatic corpus annotation, the second one is partial parsing, often referred to as *chunking*.

1.1.1 Corpus Annotation

Building a database of examples, the linguistically interpreted corpus or *treebank*, requires a lot of manual effort. We aim at automating some of the steps that are necessary during annotation, while leaving others to a human annotator. In the ideal case, the human annotator has the role of a supervisor and confirms the actions of a parser. However, we are far from this ideal situation, and the annotator frequently rejects hypotheses generated by a parser. Often, the correct analysis is not among those generated by the parser, and the human annotator needs to enter the analysis manually.

The first large and commonly available treebank was the Penn Treebank (Marcus, Santorini, & Marcinkiewicz, 1993), a collection of about 1 million words of English newspaper text (Wall Street Journal), 1 million words taken from the Brown corpus (Francis & Kucera, 1982), which consists of 15 different genres of English texts, and a subset of transliterated versions of spontaneous sentencens from the DARPA Air Travel Information System (ATIS) project. The Brown corpus, which was re-used in the Penn Treebank, was the first systematic effort to build a large text corpus

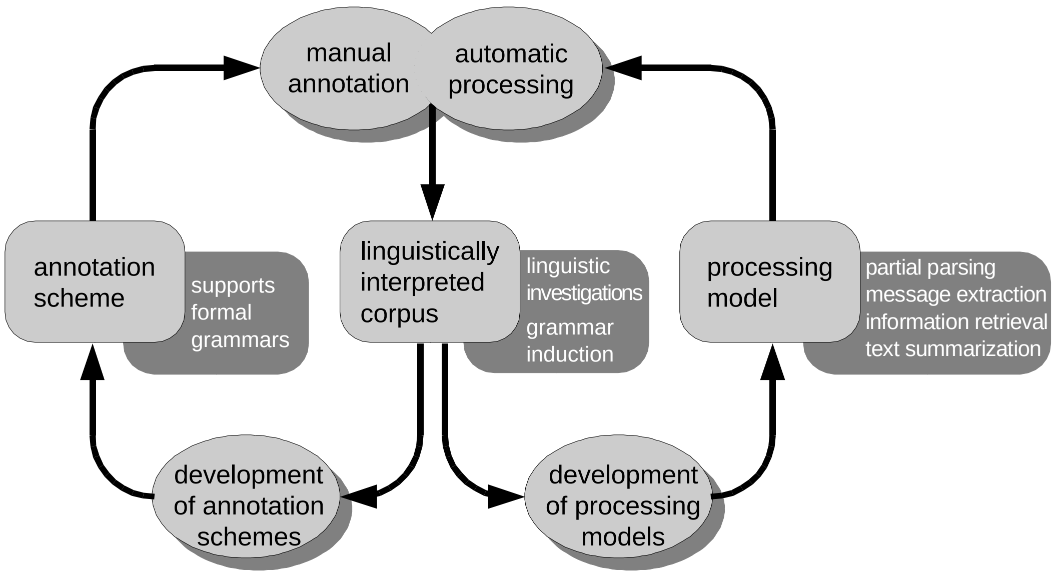


Figure 1.1: Cyclic extension and improvement of linguistic description (annotation scheme), interpreted corpus, and processing models.

annotated with syntactic categories at the word level. The corpus did not contain syntactic structures until it was added to the Penn Treebank.

The treebank offered the possibility of developing, testing and improving corpus based methods and therefore had a great influence on research in this area. Statistical models can be trained on the corpus and subsequently analyze newly presented texts.

Other English corpora followed, e.g., the Susanne Corpus (Sampson, 1995), which is a re-annotated part of the Brown corpus, and corpora of spoken language, e.g., the Switchboard corpus (Godfrey, McDaniel, & Holliman, 1993). While it was heavily disputed in the beginning, it is now more or less accepted that an interpreted corpus is a good basis for building a language understanding system. Thus, corpora for languages other than English are following, including (but not restricted to) Czech, French, German, Spanish, Chinese, and Japanese.

We think of the annotation of a corpus as a cyclic process, indicated in figure

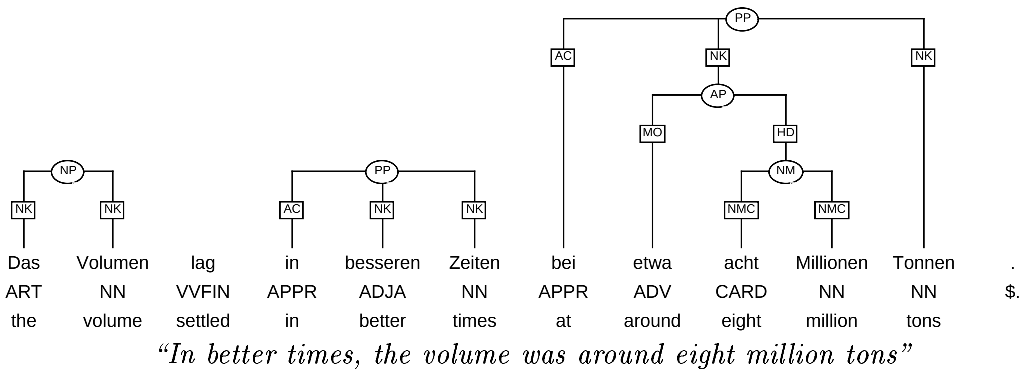


Figure 1.2: Shallow syntactic analysis (chunking)

1.1. A first version of the annotation scheme is used for annotating sentences. With feedback from the data, the annotation scheme is extended and/or changed, data is revised, new data is annotated, etc. Automatic processing methods are trained on the first annotated sentences and facilitate annotation of further sentences. Error analysis and increase of corpus size make it possible to improve automatic processing.

1.1.2 Partial Parsing and Chunking

Several applications of language technology do not depend on deep syntactic and semantic analysis. They perform their task on the basis of shallow syntactic analysis, and often an analysis as given in figure 1.2 is sufficient. This rudimentary segmentation, generated at a high speed and with high accuracy, can often be more useful and reliable than deep analysis.

Shallow and partial parsing efforts date back to the 1950's, when finite state techniques were used for large-scale parsing in the "Discourse Analysis Project." These techniques were abandoned for some time. This was partly due to the claim that finite state grammars (and even context-free grammars) are insufficient to model natural language. But it was also partly due to the lack of hardware capable of storing and processing large analyzed corpora, and the lack of suited algorithms, which were developed only recently.

Interest in shallow processing increased again in the early 1980's. Fidditch (Hindle, 1983) was one of the first partial parsers of this period. It was utilized for pre-parsing the data of the Penn Treebank. The output was subsequently manually corrected. Since then, different methods for partial parsing have been developed, using finite state automata, augmented transition networks, context-free grammars, and finite-state cascades. Partial parsing takes part in a large bootstrapping process by processing corpus data which is manually corrected and subsequently used to improve partial parsing.

Several other areas benefit from partial parsing. These include, but are not restricted to:

Message extraction, which is concerned with extraction of information relevant to a particular task from free text (who, what, when, ...) in order to fill in task-specific forms or to store the information in relational databases. This includes recognition of named entities like organizations, people, locations, etc., recognition of attributes of and relations between entities, and recognition of coreferences.

Information retrieval, the task of matching a user query against a large collection of free texts, thereby finding texts (or parts thereof) that are relevant to the query. Shallow parsing is used to extract phrases that are relevant for indexing.

Text summarization, which is the automatic generation of abstracts of variable lengths from free text. It provides systematic means to reduce the volume of a full text document without losing relevant content. The purpose of the summary is to determine the usefulness of reading the full text document.

1.2 Contribution of this Thesis

This thesis investigates tasks at different levels of syntactic natural language processing. These tasks are performed during text corpus annotation, and a high degree of automation as well as elaborate interaction between the automatic process and a human annotator are required for the efficient generation of accurate language resources. The contribution of this thesis consists of:

- The first statistical approach to the assignment of a general set of grammatical functions. These include functions like subject, direct object, head, modifier, pre- and postnominal genitive, ... Tagging grammatical functions can be seen as part-of-speech tagging at a higher level. This approach uses Markov Models to represent phrases and functions.

- The assignment of phrase categories to a given structure. This is done by an extension of the previous method. Part-of-speech tagging, assignment of grammatical functions, and assignment of phrase categories together form a complete approach to the labeling problem of a syntactic structure.

- The systematic use of alternative assignments. Their probabilities provide a measure to detect unreliable annotations and may trigger additional processing steps.

- Cascaded Markov Models. These recognize hierarchical structures by means of Markov Models. Each layer of the resulting structure is represented by a separate Markov Model. The output of a lower layer, consisting of phrase hypotheses and their probabilities, is passed as input to the next higher layer.

- The presentation of two applications of the presented methods: interactive corpus annotation, which is a new technique for efficient creation of corpus resources, and partial parsing.

- The methods are empirically tested using corpora of different languages (German and English) and different domains (newspaper text and transliterated dialogues).

The tagging and parsing techniques presented in this thesis have significantly reduced the manual effort to build the NEGRA corpus of German newspaper texts (Skut, Krenn, Brants, & Uszkoreit, 1997) and the syntactically annotated Verbmobil corpora (Stegmann & Hinrichs, 1998) of transliterated German and English spoken dialogues. More corpus initiatives have started to use these methods.

1.3 Outline

Chapter 2 introduces definitions for Markov Models and context-free grammars that will be used throughout this thesis, together with the corresponding basic algorithms.

Chapter 3 gives an overview of work that is related to the investigations of this thesis. It covers the areas of statistical part-of-speech tagging, tagging grammatical functions, stochastic context-free parsing and extensions thereof, as well as work on corpus annotation.

Chapter 4 introduces new techniques based on Markov Models that take part in the automation of corpus annotation. These techniques are: the assignment of grammatical functions, the assignment of phrase categories, and the assignment of partial structures. The processes explore several selected hypotheses in parallel in order to estimate the reliability of the top-ranked analysis and in order to make alternative assignments.

Chapter 5 presents two applications of Cascaded Markov Models. The first application is corpus annotation, the second one is partial parsing. Parsing as presented here is an extension of part-of-speech tagging, or, looking at the model, part-of-speech tagging is a special case of parsing with Cascaded Markov Models.

Chapter 6 discusses evaluation methods and presents the metrics that are used in this dissertation.

Chapter 7 reports on the evaluation of the proposed components of a partial parsing system. The corpora that are used cover the languages German and English and the domains of written and spoken language. We present results for tagging, assigning grammatical functions, assigning phrase categories and for applying Cascaded Markov Models. The final step of Cascaded Markov Models is evaluated both in the interactive annotation mode and the partial parsing mode.

Chapter 8 gives conclusions and indicates open questions and future directions.

Appendix A lists the tagsets that are used in examples throughout this thesis.

Chapter 2

Definitions

Chapter Summary

This chapter gives a short overview of the main concepts used in this thesis. It starts with introducing frequently used notations, and then describes Markov Models and stochastic context-free grammars, algorithms for processing them and algorithms for generating their parameters from annotated corpora.

2.1 Frequently Used Notations

This section introduces some notations that will be used throughout the thesis. The definitions of part-of-speech tagging, the more general labeling task, and partial parsing will use these notations.

$V = \{w_1, \ldots, w_k\}$ denotes a finite alphabet. For context-free grammars, we distinguish the set of terminal symbols V_T (usually words) and the set of non-terminal symbols V_N (usually syntactic categories of words and phrases). $V = V_T \cup V_N$.

$W = w_{i_1} \ldots w_{i_T} \in V^*$ denotes a sequence of symbols of length T. We usually leave out the second level of indices for convenience and simply write $W = w_1 \ldots w_T$.

$|W|$ denotes the length of a sequence.

$\mathcal{Q} = \{q_1, \ldots q_m\}$ denotes a finite set of states.

$Q = q_{i_1} \dots q_{i_T} \in \mathcal{Q}^*$ denotes a sequence of states of length T. We usually leave out the second level of indices for convenience and simply write $Q = q_1 \dots q_T$.

$P(X)$ denotes the probability of X.

$P(Y|X)$ denotes the conditional probability of Y given X.

$P(X \to \alpha|X)$ denotes the conditional probability of a context-free rule $X \to \alpha$ given a symbol X. This is often abbreviated as $P(X \to \alpha)$, making the conditioning implicitly.

$\max\limits_X f(X)$ denotes the maximum value of f when varying X.

$\operatorname*{argmax}\limits_X f(X)$ denotes the argument X that maximizes the function f; if more than one X maximizes f, one of them is chosen randomly.

2.2 Markov Models

2.2.1 First Order Markov Models

A discrete output, first order *Markov Model* consists of

- a finite set of states $\mathcal{Q} \cup \{q_s, q_e\}$, $q_s, q_e \notin \mathcal{Q}$, with q_s the start state, and q_e the end state;
- a finite output alphabet V;
- a set of state transitions $(q \to q')$, $q \in \mathcal{Q} \cup \{q_s\}$, $q' \in \mathcal{Q} \cup \{q_e\}$; for each transition $(q \to q')$ a probability $P(q'|q)$ is specified, the transition probability; for each state q, the sum of the outgoing transition probabilities is 1, $\sum\limits_{q' \in \mathcal{Q}} P(q'|q) = 1$;
- a set of state-output pairs $(q \uparrow w)$, $q \in \mathcal{Q}$, $w \in V$; for each pair $(q \uparrow w)$ a probability $P(w|q)$ is specified; for each state q, the sum of the output probabilities is 1, $\sum\limits_{w \in V} P(w|q) = 1$.

Figure 2.1 shows an example for a Markov Model. The Markov Model starts running in the start state q_s, makes a transition at each time step, and stops when reaching the end state q_e. The transition from one state to another is done according

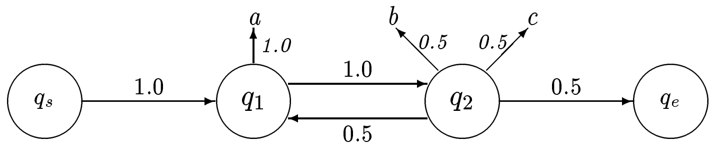


Figure 2.1: Example Markov Model. It generates the language $L = (a(b|c))^+$. The figure shows the states, outputs, transitions and probabilities for each output and transition.

to the probabilities specified with the transitions. Each time a state is entered (except the start and end state) one of the outputs is chosen (again according to their probabilities) and emitted. The Markov Model in Figure 2.1 generates the language $L = (a(b|c))^+$. The probability associated with each output string $W \in V^*$ is

$$P(W) = \begin{cases} 0.5^{|W|} & \text{if } W \text{ is of the form } (a(b|c))^+ \\ 0 & \text{otherwise} \end{cases}$$

If the state transitions depend on the previous state only, the Markov Model is of first order. If the state transitions depend on n previous states, the MM is of n-th order.

When using Markov Models for recognition, one is interested in the following questions:

1. Given a string $W \in V^*$, which sequence of states $q \in Q^*$ can have generated this string, and which is the most probable one?

2. Given a string $W \in V^*$, what is the probability of the Markov Model having generated the string?

Both problems can be solved very efficiently, i.e., in time linear to the length of the string W, $O(|W|)$. This is done by using the Viterbi algorithm (Viterbi, 1967, see section 2.2.3).

2.2.2 Higher Order Markov Models

Second, third, ..., n^{th} order Markov Model use transitional probabilities that are dependent on the previous two, three, ..., n states. These are interesting for tagging applications because states usually represent categories; so higher order Markov Model take into account larger windows of surrounding categories.

Each higher order Markov Model can be reduced to a first order Markov model that recognizes the same language and assigns the same probabilities. This is achieved by encoding different histories in different states. We encode combinations of n states of an n^{th} order Markov Model in the equivalent first order model. An n^{th} order model with k states can be represented by a first order model with k^n states.

Even though higher order Markov Model can be represented by equivalent first order models, it is often notationally more convenient to use the higher order model.

2.2.3 Dynamic Programming

The task is to calculate the probability of an output sequence $W = w_1 w_2 \dots w_T$ given a model M, $P(W|M)$. All probabilities in this section are conditioned to M, so for simplicity we write $P(W)$.

A straight-forward way to calculate the probability is to enumerate all sequences of states $Q = q_1 \dots q_T$ with length T, calculate the joint probability of the output sequence and the state sequence and sum over all state sequences, thus having

$$P(W) = \sum_{Q \in \mathcal{Q}^T} P(Q, W) = \sum_{Q \in \mathcal{Q}^T} P(Q)P(W|Q),$$

with

$$P(Q) = P(q_1|q_s)P(q_2|q_1)\dots P(q_T|q_{T-1})P(q_e|q_T)$$

and

$$P(W|Q) = P(w_1|q_1)P(w_2|q_2)\dots P(w_T|q_T).$$

The big disadvantage of this straight-forward way is the enormous computational effort that has to be made as the length of the string and the number of states grow. Since there are $|\mathcal{Q}|$ possible states which can be reached at each time, there are $|\mathcal{Q}|^T$ possible state sequences of length T, thus the computation time grows exponentially with the length of the output string.

There is a much more efficient way to calculate the probability of an output sequence, known as *dynamic programming*.

Consider the variable $\alpha_t(q)$ defined as

$$\alpha_t(q) = P(w_1 w_2 \dots w_t, q_t = q),$$

i.e., the probability of generating the partial output sequence $w_1 \dots w_t$ and being in state q at time t (given the model M).

$\alpha_t(q)$ can be expressed recursively as

$$\alpha_1(q) = P(q|q_s)P(w_1|q), \tag{2.1}$$

and

$$\alpha_{t+1}(q) = \left(\sum_{q' \in \mathcal{Q}} \alpha_t(q')P(q|q') \right) P(w_{t+1}|q), \quad \text{for } 1 \leq t \leq T-1. \tag{2.2}$$

This allows to compute

$$P(W) = \sum_{q \in \mathcal{Q}} \alpha_T(q)P(q_e|q). \tag{2.3}$$

Equation (2.1) initializes $\alpha_1(q)$ to be the joint probability of reaching state q in the first step and output w_1 in that state. Based on this initialization and the transition and output probabilities given for the model M, the subsequent α_t are calculated in equation (2.2).

The α_t are also known as the forward probabilities of the Forward-Backward Algorithm (Baum, Petrie, Soules, & Weiss, 1970).

When using α, we exploit the fact that, since there are only $|\mathcal{Q}|$ states at time t, there are again only $|\mathcal{Q}|$ states at time $t+1$, and *not* $|\mathcal{Q}|^2$, as the simple enumeration technique assumes.

The computation time needed for the dynamic programming algorithm is of the order $O(|\mathcal{Q}|^2 T)$, thus the time grows linearly with the length of the output string (opposed to an exponential growth with the straight-forward calculation).

A variant of the algorithm is used to determine the state sequence Q with the highest probability for a given output sequence W: $\underset{Q \in \mathcal{Q}}{\operatorname{argmax}} P(Q, W)$. This variant is known as the *Viterbi Algorithm* (Viterbi, 1967). The summations in equations (2.2) and (2.3) are replaced with maximizations. Instead of $a_t(q)$ we calculate $\delta_t(q)$:

$$\delta_1(q) = P(q|q_s)P(w_1|q), \tag{2.4}$$

and

$$\delta_{t+1}(q) = \left(\max_{q' \in \mathcal{Q}} \delta_t(q')P(q|q')\right) P(w_{t+1}|q), \quad \text{for } 1 \leq t \leq T-1, \tag{2.5}$$

and we have

$$\max_{Q \in \mathcal{Q}^T} P(Q, W) = \max_{q \in \mathcal{Q}} \delta_T(q)P(q_e|q). \tag{2.6}$$

Additionally, one has to keep track of the states that maximized each $\delta_t(i)$. When reaching time T, we get

$$q_T = \operatorname*{argmax}_{q \in \mathcal{Q}} \delta_T(q)P(q_e|q),$$

and by walking backwards in time, we get for the previous states

$$q_t = \operatorname*{argmax}_{q \in \mathcal{Q}} \delta_t(q)P(q_{t+1}|q), \quad t = T-1, T-2, \ldots, 1.$$

The computation time needed for the Viterbi Algorithm is again $O(|\mathcal{Q}|^2 T)$, thus linear in the length of the output string.

2.2.4 Parameter Generation

Parameters for Markov Models can be generated from annotated corpora by determining frequencies and additionally applying some smoothing technique. The first approximation for lexical and contextual parameters are relative frequencies, which are identical to maximum likelihood estimates:

Lexical probabilities:

$$\hat{P}(w|q) = \frac{f(w,q)}{f(q)} \tag{2.7}$$

Contextual probabilities (for bigrams and trigrams):

$$\hat{P}(q_2|q_1) = \frac{f(q_1,q_2)}{f(q_1)} \tag{2.8}$$

$$\hat{P}(q_3|q_1,q_2) = \frac{f(q_1,q_2,q_3)}{f(q_1,q_2)} \tag{2.9}$$

Relative frequencies cannot be used directly because they would assign zero probability to a large number of parameters that do not occur in the training corpus but are needed for test data. Therefore, a number of smoothing techniques exists that take some of the probability mass from events occurring in the training set and give it to unseen events (cf. section 3.1.3).

Parameters can also be generated from untagged corpora by using an additional, usually manually created lexicon that specifies which state can emit which output. The technique is known as Baum-Welch estimation (Baum et al., 1970). Models created from untagged corpora usually achieve worse performance than those created from tagged corpora (Elworthy, 1994).

2.3 Stochastic Context-Free Grammars

A *context-free grammar* G is a quadruple (V_N, V_T, S, R) where

V_N	is a finite set of non-terminal symbols,
V_T	is a finite set of terminal symbols; let V denote $V_N \cup V_T$,
$S \in V_N$	is a distinguished start symbol,
R	is a finite set of productions $X \rightarrow \beta$ where $X \in V_N$ and $\beta \in V^*$.

The string $\alpha X \gamma \in V^*$ can be rewritten in one step as $\alpha\beta\gamma \in V^*$ iff $X \rightarrow \beta$ is in R. This is denoted $\alpha X \gamma \Rightarrow \alpha\beta\gamma$. If a string $\phi \in V^*$ can be rewritten as the string ψ in a finite number of steps, this is denoted $\phi \Rightarrow^* \psi$. $L(G)$ denotes the set of strings $W \in V_T^*$ (the *language*) generated by G and is defined as $\{W \in V_T^* : S \Rightarrow^* W\}$.

A *derivation* of terminal sequence $W \in V_T^*$ is the sequence of rewrites $S = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \ldots \Rightarrow \alpha_k = W$. The *leftmost derivation* of W is the derivation that rewrites the leftmost non-terminal symbol in each step. A *parse tree* of W is the tree representation of a leftmost derivation, i.e., the root is labeled S, the leafs are labeled with elements of V_T such that the yield of the tree is W, and all internal nodes are labeled with elements of V_N such that they reflect the rewrites of the derivation. A grammar G is *finitely ambiguous* iff there is a finite number of leftmost derivations for any element in $L(G)$. This is equivalent to requiring that $X \Rightarrow^+ X$ is impossible for any $X \in V_N$.

A *stochastic context-free grammar* G is a quintuple (V_N, V_T, S, R, P) where

V_N	is a finite set of non-terminal symbols,
V_T	is a finite set of terminal symbols; let V denote $V_N \cup V_T$,
$S \in V_N$	is a distinguished start symbol,
R	is a finite set of productions $X \rightarrow \beta$ where $X \in V_N$ and $\beta \in V^*$.
P	is a function from R to $[0,1]$ such that: $\forall X \in V_N : \sum_{\beta \in V^*} P(X \rightarrow \beta) = 1$

The probability of a derivation $P(S \Rightarrow^* W)$ is defined as the product of the rules' probabilities that are used in the derivation. The probability of a string is defined as the sum of the probabilities of all leftmost derivations that yield the string.

2.3.1 Chart Parsing

The Viterbi algorithm can be adapted to parsing with stochastic context-free grammars. We present a stochastic variant of the Cocke-Younger-Kasami (CYK) algorithm (Aho & Ullman, 1972). Assume that $V_N = \{X_1, \ldots, X_N\}$, S is the start symbol, and $W = w_1 \ldots w_T \in V_T^*$. The algorithm utilizes a set of accumulators

$$\delta_{s,t}(X_i) \qquad 0 \le i \le N, 0 \le s < t \le T$$

in order to parse the sequence W. These are defined as the maximum probability of any partial parse tree spanning the substring $w_{s+1}, \ldots, w_t$. The probability of the most probable parse tree for W is thus $\delta_{0,T}(S)$.

The parse tree is constructed bottom-up. The basic algorithm assumes the grammar G to be in Chomsky Normal Form.

Initialization:

$$\delta_{t-1,t}(X_i) = P(X_i \to w_t) \qquad 1 \le i \le N, 1 \le t \le T \tag{2.10}$$

Recursion:

$$\delta_{r,t}(X_i) = \max_{j,k,r<s<t} P(X_i \to X_j X_k)\delta_{r,s}(X_j)\delta_{s,t}(X_k) \qquad 1 \le i \le N, 0 \le r < t \le T \tag{2.11}$$

Termination:

$$P(W|G) = \delta_{0,T}(S) \tag{2.12}$$

The arguments that maximized each $\delta_{r,t}(X_i)$ are stored, so we can generate the most probable parse by an additional processing step after computing $\delta_{0,T}(S)$.

The algorithm can be generalized to the case of a finitely ambiguous stochastic context-free grammar, which is not necessarily in Chomsky Normal Form (CNF). This makes the formulas less straight-forward, but the algorithm is better suited to linguistic applications, which usually need structures different from CNF.

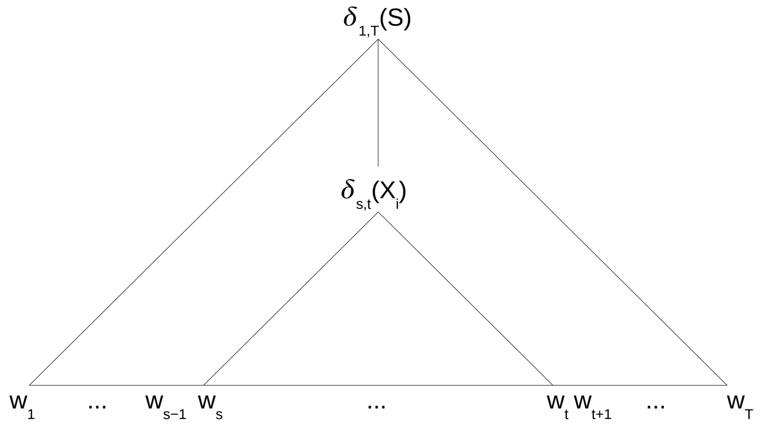


Figure 2.2: Accumulators $\delta_{s,t}(X_i)$ recursively store the maximum probability of partial parse trees having X_i as root and spanning $w_s \dots w_t$. Finally, $\delta_{1,T}(S)$ is the maximum probability of all parse trees spanning the complete terminal sequence.

Recursion for a finitely ambiguous context-free grammar:

$$\delta_{r,t}(X_i) = \max_{\substack{(X_i \to \alpha) \in R \\ \alpha = Y_1^{s_0,s_1} \dots Y_k^{s_{k-1},s_k} \\ s_0 = r, s_k = t}} P(X_i \to \alpha) \prod_{j=1}^{k} \delta_{s_{j-1},s_j}(Y_j) \tag{2.13}$$

$1 \leq i \leq N, 0 \leq r < t \leq T$. The superscript in $Y_j^{s_{j-1},s_j}$ indicates that the symbol $Y_j \in V$ is rewritten to terminals from position $s_{j-1}+1$ to s_j. This requires that the domain of $\delta_{r,t}$ is extended to terminal symbols, and we get the *initialization for a finitely ambiguous CF grammar* as:

$$\delta_{t-1,t}(Y_i) = \begin{cases} 1 & \text{if } Y_i = w_t \\ 0 & \text{if } Y_i \neq w_t \end{cases} \qquad 1 \leq t \leq T, Y_i \in V_T \tag{2.14}$$

The time complexity to find the most probable parse of an input string W according to $G = (V_N, V_T, S, R, P)$ is of the order $O(|V_N|^3 |W|^3)$.

For a more detailed description of stochastic context-free grammars see e.g., (Krenn & Samuelsson, 1997).

2.3.2 Parameter Generation

We consider parameter generation from annotated corpora (as opposed to using raw corpora together with some initial guess for the parameters). Probabilities for context-free rules can be estimated directly from annotated structures of sufficient size by using relative frequencies:

$$\hat{P}(X \to \alpha) = \frac{f(X \to \alpha)}{f(X)} \tag{2.15}$$

In this thesis, we will only investigate training on annotated corpora. Training on unannotated corpora can be done with the inside-outside algorithm. For an introduction into this topic see e.g., (Krenn & Samuelsson, 1997).

Usually, the model needs additional smoothing in order to account for cases in which no single, contiguous structure can be assigned or previously unseen rules are needed to generate a structure.

A simple method to determine best partial parses if no complete parse is possible is to add a new non-terminal symbol X^{new} and all rules of the form

$$X^{new} \to YZ, \quad Y, Z \in V_N \cup \{X^{new}\}$$

to the grammar. If $\delta_{0,T}(S) = 0$, then $\delta_{0,T}(X^{new})$ and the arguments that maximized the δ's are used to determine the best partial parses. Probabilities of the new rules are either all set to the same value, summing up to 1, or are weighted according to the frequency of the involved symbols.

Another possibility is to add all rules that did not occur in the corpus, using only existing non-terminal symbols, and assign small probabilities to them. This is only practical for grammars in CNF with a small number of non-terminals, because the number of rules increases drastically.

Chapter 3

Related Work

Chapter Summary

We give an overview of work related to the topics investigated in this thesis. These are current approaches in statistical part-of-speech tagging, assigning grammatical functions, statistical parsing, partial parsing, and corpus annotation.

3.1 Part-of-Speech Tagging

The task of part-of-speech (PoS) tagging is the unique annotation of a word with a syntactic category, called *part-of-speech* or *tag*. Different methods have been developed to perform this task. They all have in common that they exploit knowledge about the words and a small context in which the words appears. But the means by which this knowledge is exploited differ. The main paradigms for part-of-speech tagging are:

statistical: Transitional probabilities between tags and lexical probabilities of tags for words are used. The process finds the sequence of tags that has the highest probability given a sequence of words (Church, 1988; DeRose, 1988; Cutting, Kupiec, Pedersen, & Sibun, 1992; Kupiec, 1992; Weischedel, Meteer, Schwarz, Ramshaw, & Palmucci, 1993; Merialdo, 1993; Brants & Samuelsson, 1995; Ratnaparkhi, 1996, and many more).

transformation based: In a first stage, a dumb tagger assigns first guesses for tags to the words. This is often done by a unigram tagger. The second stage applies finite state rules that are learned from a corpus and corrects the tagging errors of the dumb tagger (Brill, 1993).

finite state: A first stage makes a lexicon lookup for each word and assigns all tags found by this lookup. A second stage employs rules for removing tags of ambiguous words, leaving the one that is correct in the particular context. The rules are encoded as finite state automata or finite state transducers. They are either manually written (Roche, 1992; Silberztein, 1993, 1997; Tapanainen & Voutilainen, 1993; Voutilainen, 1994) or generated from a corpus: Roche and Schabes (1995) show the equivalence of transformation-based tagging and tagging with finite state transducers and thereby provide an efficient processing method. (Kempe, 1997) approximates HMMs with finite state transducers.

memory based: Combinations of words and their context are extracted from a corpus and stored, either directly or in a decision tree. In the tagging phase, the closest match within the training data is searched in order to determine the assigned tag (Daelemans, Zavrel, Berck, & Gillis, 1996).

The best results that are reported in the literature are those for hand-coded rules. Comparison of the other, automatically trained systems yield the following results. If the different paradigms are compared, statistical taggers yield the best results (Volk & Schneider, 1998; Halteren, Zavrel, & Daelemans, 1998). If combinations of systems (i.e., integration of systems that implement different paradigms) are compared, best results are obtained by a combined system that incorporates a strong statistical component (Halteren et al., 1998).

This thesis is concerned with statistical part-of-speech tagging. Our intention is to statistically model other sequential processes using the same or similar techniques.

Let $\mathcal{T}$ be defined as the set of all tags, and V the set of all words. In a statistical tagging task, one is given a sequence of words $W = w_1 \dots w_k \in V^*$, and is looking for a sequence of tags $T = t_1 \dots t_k \in \mathcal{T}^*$ that maximizes the conditional probability $P(T \mid W)$; hence one is looking for

$$\underset{T}{\operatorname{argmax}} \quad P(T|W) = \underset{T}{\operatorname{argmax}} \frac{P(T)\ P(W|T)}{P(W)}.$$

$P(W)$ is independent of the chosen tag sequence, so it is sufficient to find

$$\underset{T}{\operatorname{argmax}} \;\; P(T)\; P(W|T).$$

In an n-gram model for each pair $(w,t) \in V \times \mathcal{T}$, the lexical probabilities

$$P(w \mid t),$$

and for each n-tuple $(t_1 \dots t_n) \in \mathcal{T} \times \dots \times \mathcal{T}$ the transition probabilities

$$P(t_n \mid t_1 \dots t_{n-1})$$

are defined. These approximate the lexical and conditional probabilities with

$$P(W|T) \approx P(w_1|t_1)\; P(w_2|t_2)\; \cdots\; P(w_k|t_k)$$

and

$$\begin{aligned} P(T) &= P(t_1)\; P(t_2|t_1)\; P(t_3|t_1,t_2)\; \cdots\; P(t_k|t_1 \dots t_{k-1}) \\ &\approx \prod_{i=1}^{k} P(t_i \mid t_{i-n+1} \dots t_{i-1}) \end{aligned}$$

Note that the beginning of the string requires some extra handling. Additional tags $t_{-n+2} \dots t_0$ are introduced, marking the "start of string" position, or, when using Markov Model terms, initial states are introduced.

Now the joint probability of a string of words $W = w_1 \dots w_k$ having a string of tags $T = t_1 \dots t_k$ is the product of their lexical and transition probabilities

$$P(W,T) = P(T)P(W|T) \approx \prod_{i=1}^{k} P(t_i \mid t_{i-n+1} \dots t_{i-1}) P(w_i \mid t_i).$$

(making the Markov assumption) and finding the best string of tags T for a given string of words W is done by finding

$$\underset{t_1 \dots t_k}{\operatorname{argmax}} \prod_{i=1}^{k} P(t_i \mid t_{i-n+1} \dots t_{i-1}) P(w_i \mid t_i).$$

This formula describes a part-of-speech n-gram model. The best compromise between the size of the corpus that is needed for parameter estimation and the quality of the output are usually trigram models, having $n = 3$.

3.1.1 n-gram Models

n-gram models are identical to Markov Models of order $(n-1)$. This means, to determine the state for the current word, the states of the previous $(n-1)$ words are taken into account. Additionally, the states have a fixed meaning. A state either represents a single word (*word n-grams*), or a syntactic category or part-of-speech (*PoS n-grams*).

For word-n-grams, a state can emit exactly one word with probability 1, all other words are emitted with probability 0.

For PoS-n-grams, a state emits words belonging to the represented category with a probability greater than 0, and all other words with probability 0. Here, the output probabilities $P(w|q)$ are called *lexical probabilities*, and the transition probabilities $P(q_n|q_1,\ldots,q_{n-1})$ are called *contextual probabilities*.

Generally speaking, word n-grams are a special case of PoS n-grams, where different words belong to different categories and each word constitutes a separate category.

n-gram taggers have been applied successfully for several years and reach a level of accuracy of 95–97% for English and German texts (cf. section 3.1.5).

3.1.2 Estimating Parameters

One problem in using n-gram models is the estimation of parameters, i.e., determining the output and transition probabilities. In the following, the case of PoS-n-grams is considered.

Let $\mathcal{T}$ be the set of categories. $\mathcal{T}$ is isomorphic to the set of states $\mathcal{Q}$ for the Markov Model[1]. Let V be the output alphabet. Then the lexical probabilities $P(w_i|t_i)$, $w_i \in V, t_i \in \mathcal{T}$, and the contextual probabilities $P(t_i|t_{i-n+1}\ldots t_{i-1})$, $t_j \in \mathcal{T}$, must be determined.

This can be done by evaluating a sufficiently large corpus, which is already annotated. The count (*frequency*) in the corpus for each pair $(w,t) \in V \times \mathcal{T}$

$$f(w,t)$$

[1]Except start and end states, for which there are usually no corresponding tags.

and for each n-tuple $(t_1 \dots t_n) \in \mathcal{T} \times \dots \times \mathcal{T}$

$$f(t_1, \dots, t_n)$$

is determined. Then the probabilities are estimated using the Maximum Likelihood Estimation (MLE) by

$$\hat{P}(w_i|t_i) = \frac{f(w_i, t_i)}{\sum\limits_{w \in V} f(w, t_i)}$$

and

$$\hat{P}(t_i|t_{i-n+1} \dots t_{i-1}) = \frac{f(t_{i-n+1} \dots t_i)}{\sum\limits_{t \in \mathcal{T}} f(t_{i-n+1} \dots t_{i-1} t)}$$

This maximizes the probability of observing the specific frequency given the estimated probability $\hat{P}$.

Additionally, there exist parameter estimation methods that do not require a previously (in most cases manually or semi-automatically) annotated corpus, e.g., the Baum-Welch estimation method (Baum et al., 1970). Yet, training on annotated corpora generally yields better results than using unannotated corpora (Elworthy, 1994).

3.1.3 The sparse data problem

n-gram models need a corpus to be "trained" on, i.e., the parameters are estimated from frequency counts in the corpus. But even with very large corpora there is the *sparse data problem*: the fact that a lot of the frequencies used for estimation of context probabilities are zero ($f(t_1 \dots t_n) = 0$ for many of the $(t_1 \dots t_n) \in \mathcal{T}^n$). This has a very undesirable effect. If a string of tags $t_1 \dots t_k, k \geq 1$, contains a substring of tags $t_l \dots t_m, 1 \leq l \leq m \leq k$, that has zero probability, the complete sequence $t_1 \dots t_k$ is assigned zero probability:

$$P(t_l \dots t_m) = 0 \quad \Longrightarrow \quad P(t_1 \dots t_{l-1} t_l \dots t_m t_{m+1} \dots t_k) = 0$$

Thus, all sequences are assigned the same probability regardless of the instances of $t_1 \dots t_{l-1}$ and $t_{m+1} \dots t_k$. This is not only intuitively wrong, but also yields very poor results in empirical validations.

Therefore, several methods to avoid zero probabilities are suggested in the literature. These are:

- The Expected Likelihood Estimator (ELE) (see e.g. Gale & Church, 1990). All frequencies (including zero frequencies) f are replaced by $f^* = f + 0.5$. The new frequencies f^* are used for maximum likelihood estimation.
- The Minimax Method (Steinhaus, 1957). All frequencies f are replaced by $f^* = f + \sqrt{N}/2$, where N is the size of the sample.
- The Good-Turing Method (Good, 1953). All frequencies f are replaced by $f^* = (f+1)N_{f+1}/N_f$, where N_f denotes the frequency of frequency f. Katz (1987) combines this method with a back-off model.
- Linear Interpolation (Jelinek & Mercer, 1980; Brown, Pietra, deSouza, Lai, & Mercer, 1992). All trigram probabilities $P(t_3|t_1,t_2)$ are estimated by

$$\begin{aligned} P(t_3|t_1,t_2) &= \lambda_1(t_1,t_2)\hat{P}(t_3) \\ &\quad +\lambda_2(t_1,t_2)\hat{P}(t_3|t_2) \\ &\quad +\lambda_3(t_1,t_2)\hat{P}(t_3|t_1,t_2), \end{aligned}$$

  where $\hat{P}$ denotes maximum likelihood probabilities. Thus, a linear combination of uni-, bi-, and trigram probabilities is used. Using two categories as context and estimating different λ's for each pair of categories is usually too fine-grained and leads to sparse data problems when estimating the weights. The solution is to use a small number of equivalence classes of contexts instead. Using just one equivalence class leads to the context-independent version which estimates the weights λ_i independently of t_1 and t_2.
- Linear abstraction (Samuelsson, 1996). The method uses a sequence of increasingly general contexts $C_m \subset C_{m-1} \subset \ldots \subset C_0$. Probability estimates are recursively defined by using the relative frequency r and the estimate of the next general context:

$$P^*(x|C_k) = \frac{r(x|C_k) + \theta P^*(x|C_{k-1})}{1+\theta}$$

using a weight θ and the initialization

$$P^*(x|C_0) = r(x).$$

The weight θ is estimated by using the standard deviation of probabilities for context C_k. For trigram part-of-speech tagging, the sequence of increasingly more general contexts consists of trigrams (C_2), bigrams (C_1), and unigrams (C_0).

3.1.4 Handling of Unknown Words

Currently, the best method of handling words of inflected languages that are not in the lexicon is a suffix analysis as proposed in (Samuelsson, 1993). Tag probabilities are set according to the word's ending. The suffix is a strong predictor for word classes, e.g., words in the Wall Street Journal part of the Penn Treebank ending in *able* are adjectives (JJ) in 98% of the cases (e.g. fashionable, variable) , the rest of 2% are nouns (e.g. cable, variable).

The probability distribution is generated from words in the lexicon sharing the same suffix of some predefined maximum length. Probabilities are smoothed by an instance of linear abstraction (see section 3.1.3). It calculates the probability of a tag T given the last m letters l_i of an n letter word: $P(T|l_n, \ldots, l_{n-m+1})$. The sequence of increasingly more general contexts omits more and more characters of the suffix, such that $P(T|l_n, \ldots, l_{n-m+2})$, $P(T|l_n, \ldots, l_{n-m+3})$, $\ldots$, $P(T)$ are used for smoothing.

3.1.5 Implementations

A number of existing implementations of statistical taggers are described in the literature. Probably the first statistical tagger was CLAWS (Marshall, 1983). It already incorporated lexical and contextual probabilities learned from a tagged corpus. But it did not exploit the advantages of dynamic programming and therefore used an exponential algorithm.

The first efficient n-gram taggers using (variations of) the Viterbi algorithm were the Church tagger (Church, 1988) and VOLSUNGA (DeRose, 1988). Both taggers train on tagged corpora and integrated simple methods of smoothing. Cutting et al. (1992) presented the XEROX tagger that was trained on a lexicon and untagged corpora. A more recent tagger uses the maximum entropy framework for parameter estimation (Ratnaparkhi, 1996). Halteren et al. (1998) compared different taggers for English. Best results (around 97%) were achieved by a combination of systems.

These taggers were used to process English texts. The same techniques were applied to other languages. First results for German were reported by Wothke, Weck-Ulm, Heinecke, Mertineit, and Pachunke (1993). Several other investigations and implementations followed, e.g., (Schmid, 1995; Steiner, 1995; Armstrong, Russell, Petitpierre, & Robert, 1995; Schütze, 1995; Lezius, 1996) and our work (Brants, 1996a). They report accuracies ranging from 93% to 97% for tagsets of 33 – 56 tags.

3.2 Assignment of Grammatical Functions

Grammatical functions as used here denote relations between a node in a parse tree and its immediately dominating node. Examples for such functions are *subject*, *object*, *head*, *modifier*, etc. The assignment of grammatical functions is relatively new to the areas of statistical and finite-state processing, although most traditional grammars incorporate this or very similar concepts.

Apart from the approach presented in this thesis, only one other stochastic approach to assigning grammatical functions can be found in the literature. DeLima (1997) developed a method for the distinction between subject and direct object in German for those sentences that contain one verb v and two nominative/accusative NPs with head nouns n_1 and n_2. The decision is based on the triple (n_1, v, n_2) and frequencies $f(n_1, v, n_2, subj = 1)$ that are taken from a corpus ($subj = 1$ indicates that the first NP is the subject; the alternative is $subj = 2$). Additionally, Katz' back-off model (Katz, 1987) is used in order to handle the sparse data problem that arises when using frequencies based on triples of words. The frequencies are backed off by using $f(n_1, v, subj = 1)$, $f(n_1, v, subj = 2)$, $f(v, subj = 1)$, and $f(v, subj = 2)$, which means that the identity of the second noun is ignored as a first step in the back-off, and the identities of the first and second noun are ignored as a second step.

The advantage of this approach is that the model is trained on a raw corpus with the help of a morphological component and a shallow parser. But the approach is restricted to only two grammatical functions (*subject* and *direct object*), and the strategy used cannot be straightforwardly generalized to other functions.

3.3 Stochastic Natural Language Parsing

3.3.1 Context-Free Parsing

The basic parsing schemes for stochastic context-free grammars as presented in section 2.3 are subject to several investigations and extensions.

One important addition is the use of probabilistic context. Instead of

$$P(X \to \alpha | X)$$

a context-sensitive probability model uses

$$P(X \to \alpha | Y \to \beta X \gamma)$$

so that Y is the parent node of X. This can be extended to an arbitrary amount of context.

Magerman and Weir (1992) use the parent production and part-of-speech trigrams to condition rule probabilities. Edges that are proposed at some point in the chart receive a probability according to

$$P(A \to \alpha | C \to \beta A \gamma, a_0 a_1 a_2)$$

where C is the non-terminal immediately dominating A, a_1 is the part-of-speech of the leftmost word of A, a_0 is the part-of-speech to the left of a_1, and a_2 is the part-of-speech to the right.

Black et al. (1993) present a history-based model that theoretically takes into account the complete parsing history up to the current point. For practical reasons, they restrict the history to the path from the current node to the root of the tree according to its leftmost derivation. They augment this technique with a decision tree that examines the path and selects distinctive information.

Jelinek et al. (1994) regard the complete derivation history as useful. Due to sparse data problems, the history is restricted to a five node window around the node in question. The different histories are classified by decision trees.

These types of probability models for grammars fall in the class of *history based grammars*. They are always reported to be superior to a standard probabilistic context-free model. A general approach to parsing with history based grammars and decision trees is presented by Magerman (1995). He achieves precision and recall rates of 86.3% and 85.8% on the Wall Street Journal part of the Penn Treebank.

Pereira and Schabes (1992) investigate parameter estimation for probabilistic context-free grammars from annotated and unannotated corpora. Both yield comparable results as far as cross-entropy of the derived model and the training corpus is concerned. But using an (at least partially) annotated corpus yields far better bracketing accuracy.

Sekine developed and implemented a bottom-up probabilistic chart parser which finds the parse tree with a best-first search algorithm (Sekine & Grishman, 1995; Sekine, 1998). The underlying English grammar is semi-context-sensitive with two non-terminals. It was automatically induced from the Penn Treebank. Recall and precision for Penn Treebank data is reported to be up to 75.2%/79.6% for the best version of the parser.

Context-free rules represent sub-trees of depth 1 in a context-free structure. Bod and Scha (1994) and Bod (1993, 1995) extend this notion and use sub-trees of arbitrary depth for parsing. They generate all sub-trees and their frequencies from an annotated corpus and uses them as rules for a grammar. Figure 3.1 shows all sub-trees for a corpus consisting of just the one sentence *John likes Mary.*

The advantage of Data Oriented Parsing is the variable depth of the sub-trees which adapt better to linguistic constructs than structures that are restricted to just one level as in context-free rules. The idea is to have large sub-trees for relatively fixed constructs, and small sub-trees for elements that can be arbitrarily combined.

A big problem of this approach is the large number of different sub-trees occurring in a corpus. The number grows exponentially with the size of the corpus. This can be partly solved by restricting the depth of the sub-trees to a maximum depth n

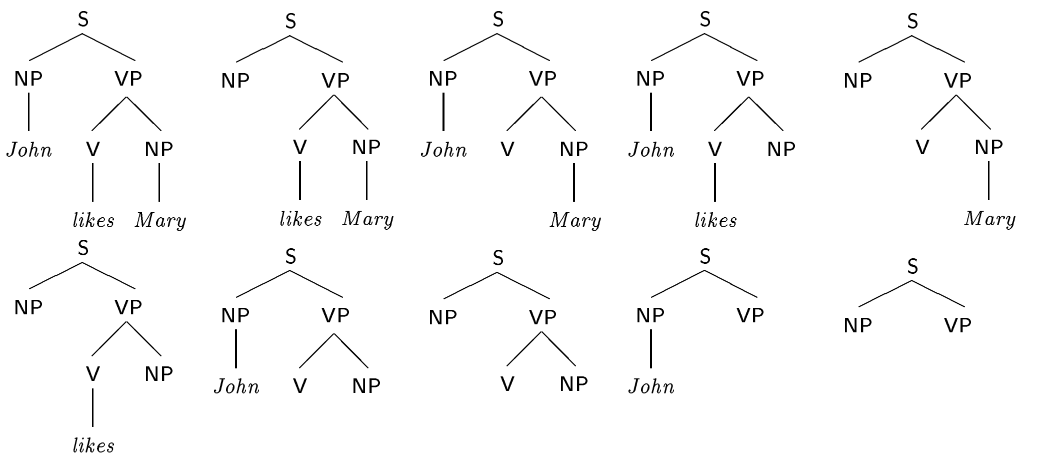


Figure 3.1: Data oriented parsing uses each tree occurring in the corpus and all sub-trees together with their frequencies for parsing (Bod, 1995).

(e.g., $n = 5$). But the restriction gives away part of the advantage of Data Oriented Parsing.

The second problem is that finding the most probable parse given a sentence is non-polynomial (Sima'an, 1996a). Nevertheless, the time complexity of finding the most probable derivation is $O(l^3)$ for a sentence of length l. Sima'an (1996b) presents an improved and efficient parsing algorithm for finding the most probable derivation. Goodman also presents efficient parsing algorithms for DOP (Goodman, 1996, 1998).

Ratnaparkhi (1997) adds a new parameter estimation method, maximum entropy modeling, in order to better handle sparse data. Additionally, he uses a special process during parsing that assigns tags indicating beginning, continuation and end of phrases in order to facilitate processing of a stochastic context-free parser. He achieves very good parsing results and reports 85.6% recall and 86.8% precision on Penn Treebank data.

Furthermore, there are investigations on parsing context-free structures using dependencies between words. Collins (1996) uses lexicalization and models of dependencies to achieve robust parsing with high accuracy. The model is extended by using the distinction of complements and adjuncts and the treatment of traces and wh-movement (Collins, 1997). Eisner (1996) presents three probabilistic parsing models using dependencies between words and selectional preferences. Link Grammar (Lafferty, Sleator, & Temperley, 1992; Sleator & Temperley, 1993) is closely related to dependency grammar. Words in a parsed sentence are connected by typed edges. Grinberg, Lafferty, and Sleator (1995) present a robust parser for the model.

3.3.2 Propagating Lexical Information

A problem of stochastic context-free grammars that are learned from a treebank is the rather small amount of information contained in non-terminal nodes. The Penn Treebank, for example, uses just one category label for noun phrases: NP. When generating a grammar from the treebank, one finds rules like

$$\mathsf{S} \rightarrow \mathsf{NP}\ \mathsf{VP}$$

$$\mathsf{NP} \rightarrow \mathsf{NP}\ \mathsf{VP}$$

as they occur in

$[_{S}\ [_{NP}$ *authorities* $]\ [_{VP}$ *released television footage to Western news agencies* $]]$.

and

. . . *according to* $[_{NP}\ [_{NP}$ *government figures* $]\ [_{VP}$ *released yesterday* $]]$.

The categories NP and VP are not informative as to whether their parent node should be S or NP. One source for disambiguation, the context, is exploited: the top-level node is preferably of category S, while *according to* prefers a following NP. But the other source, i.e., the internal elements, is not used, although it is a strong indicator in these cases, since the alternative analyses

* $[_{NP}\ [_{NP}$ *authorities* $]\ [_{VP}$ *released television footage to Western news agencies* $]]$

and

* $[_{S}\ [_{NP}$ *government figures* $]\ [_{VP}$ *released yesterday* $]]$.

are not possible in (almost) any context. The mere encoding of a category in a non-terminal node is generally insufficient.

Traditional phrase-structure grammars solve this problem by encoding properties of sub-structures as features and use unification to ensure that different parts of a structure match. If these features were encoded in a treebank, one could probably use them for parsing. However, a large number of features is necessary for traditional unification grammars, and these cannot be found in current treebanks.

A solution to this problem that does not require additional manual coding effort is the propagation of lexical information. Each non-terminal node is associated with a lexical representative. This association is recursively defined. The lexical representative of a phrasal node is the lexical representative of one of its constituents. Usually, a *head* element is chosen to propagate lexical information. The notion of "head" does not necessarily correspond to that of any larger grammar theory, but is only loosely related. For each type of phrase, the head is defined to be a prominent word in the phrase. As an example, the head of a noun phrase is usually the rightmost noun in the phrase. It is generally assumed that the *exact* definition of "head" only has marginal influence on the performance of a stochastic parsing model.

When propagating lexical information, the parser distinguishes rules like

$$\mathsf{S}\text{-}\textit{released} \;\rightarrow\; \mathsf{NP}\text{-}\textit{authorities}\;\; \mathsf{VP}\text{-}\textit{released}$$

and

$$\mathsf{S}\text{-}\textit{released} \;\rightarrow\; \mathsf{NP}\text{-}\textit{figures}\;\; \mathsf{VP}\text{-}\textit{released}$$

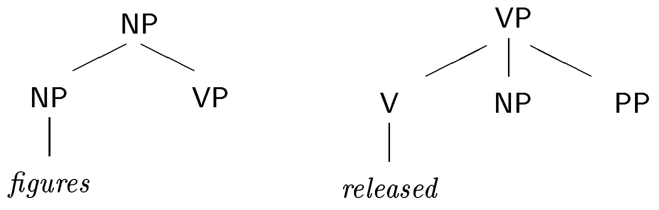
of which the latter is assigned a much lower probability according to the corpus. This type of propagation increases the number of context-free rules and therefore needs a good model for smoothing in order not to run into sparse data problems. One can use back-off smoothing (Katz, 1987), successive abstraction (Samuelsson, 1996), the maximum entropy approach (Berger, Della Pietra, & Della Pietra, 1996; Ratnaparkhi, 1997), and others.

Propagation of lexical information is related to lexicalization in the Tree Adjoining Grammar Framework (Schabes, Abeillé, & Joshi, 1988). There, a lexicalized grammar systematically associates each structural element with a lexical anchor.

Schabes and Waters (1993) extend this notion of lexicalization to context-free grammars. Instead of a rule like

$$\mathsf{NP} \rightarrow \mathsf{NP}\ \mathsf{VP} \quad \text{and} \quad \mathsf{VP} \rightarrow \mathsf{V}\ \mathsf{NP}\ \mathsf{PP}$$

they use larger sub-trees for parsing. Each sub-tree is associated with a lexical item, e.g.,



Johnson (1998) varied the way of encoding structures and thereby heavily influenced parsing accuracy when a treebank is used to induce a probabilistic grammar. The same is true for the variation of information in the node labels.

3.4 Partial Parsing

3.4.1 Tagging with Structural Tags

Early work on parsing with structural tags started with the recognition of simple NP boundaries (Church, 1988). The approach uses a transition matrix that indicates the probability that an NP starts or ends between these two elements.

Similar approaches can recognize a larger set of structural categories. Joshi and Srinivas (1994) use part-of-speech tagging techniques to assign elementary trees in the Tree Adjoining Grammar (TAG) framework to each word. The technique is named "Supertagging" and used as a preprocessing step for a full parser in order to reduce ambiguity.

Brants and Skut (1998) show that seven simple structural tags are sufficient to reliably recognize complex NPs and PPs in German. The tags encode the hierarchical position of a word in syntactic structure relative to the preceding word. Skut (forthcoming) further develops this technique. He investigates and optimizes the type and amount of categorial information encoded in the chunk tags. The model is combined with the maximum entropy parameter estimation to exploit different features and to improve smoothing, and it is combined with postprocessing filters in order to eliminate systematic errors. The combined model yields a high performance chunking system.

These three techniques have in common that they encode a finite number of partial structures and assign one of them to each word, depending on the local context.

The ENGCG Parser (Karlsson, Voutilainen, Heikkilä, & Anttila, 1994) also produces a shallow analysis, but based on hand-crafted rules. The parser encodes morphological information, part-of-speech, and some types of syntactic functions as well as NP boundaries at the word level.

Ramshaw and Marcus (1995) presented an approach to transformation based chunking. They exploit a technique that was originally developed to learn rules for part-of-speech tagging. Chunking approaches in the memory based framework with tags marking boundaries were presented in (Argamon, Dagan, & Krymolowski, 1998) and (Veenstra, 1998).

3.4.2 Finite State Cascades

Parsing with finite state cascades utilizes a series of finite state transducers that operate at different levels of a syntactic structure and recognize non-recursive structures. The output of a transducer at a lower level is used as input for the next higher level. Phrases never contain other phrases from the same or higher levels. Figure 3.2 shows a parse tree represented as a sequence of levels.

The systems usually specialize each level to recognize particular elements of a structure, e.g., proper nouns, date/time expressions, simple NPs, PPs, PP attachment, domain specific events, etc. Finite state cascades require a specially designed grammar because of the restriction that phrase types recognized at lower levels cannot contain phrase types recognized at higher levels. The sets of transducers are manually built, usually in a process that alternates construction and testing on a corpus.

Implementations of finite state cascades can be dated back to Joshi (1960) and

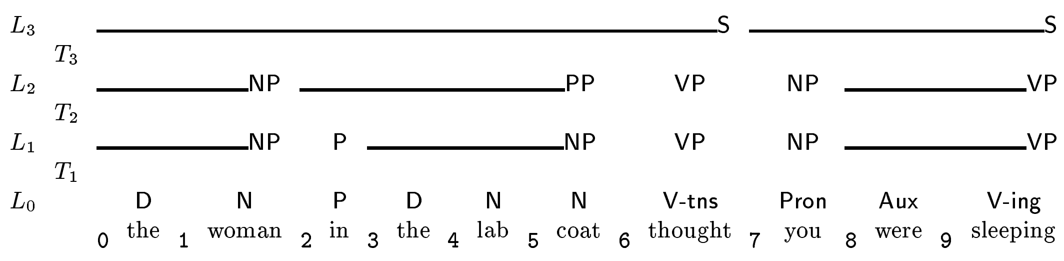

L_3
S
S
T_3
L_2
NP
PP
VP
NP
VP
T_2
L_1
NP
P
NP
VP
NP
VP
T_1
L_0
D N P D N N V-tns Pron Aux V-ing
0 the 1 woman 2 in 3 the 4 lab 5 coat 6 thought 7 you 8 were 9 sleeping

Figure 3.2: Finite state cascades. Transducer T_i operates on phrases or tags at level L_{i-1} and emits phrases at level L_i (Abney, 1996). L_0 is produced by a part-of-speech tagger.

Harris (1962) who describe an early attempt at large-scale parsing in the Discourse Analysis Project. They use the UNIVAC-1 parser that mainly consists of a cascade of finite state transducers. This parser was recently reconstructed (Joshi & Hopely, 1997). It already incorporates several state-of-the-art techniques for parsing text corpora.

CASS (Abney, 1990, 1991, 1996) is a partial parser that tags its input with a trigram part-of-speech tagger. Subsequently, finite state transducers recognize non-recursive basic phrases (chunks). Each transducer emits a single best analysis that serves as input for the transducer at the next higher level.

FASTUS (Appelt, Hobbs, Bear, Israel, & Tyson, 1993) is heavily based on pattern matching. Each pattern is associated with one or more trigger words. It uses a series of non-deterministic finite-state automata to build chunks; the output of one automaton is passed as input to the next automaton.

The Saarbrücker Message Extraction System (SMES Declerck, Klein, & Neumann, 1998) combines several shallow processing modules, including a tokenizer, morphological analyzer, part-of-speech tagger and shallow parsing.

Roche (1994) presents a parsing method which uses the fix point of a finite-state transducer. The transducer is iteratively applied to its own output until the output remains identical to the input. Although the fix point of a finite state transducer is Turing equivalent, the method can be successfully used for efficient processing with large grammars.

Gross (1997) presents the construction of *local grammars.* Manually written rules are expressed as finite state automata. These capture local constraints. The automata can be combined to large coverage and lexicalized grammars. The grammar is written directly in the finite-state framework.

(Roche, 1997) assumes that the grammar is already given and presents techniques for parsing context-free grammars with finite-state transducers. Although finite-state transducers cannot exactly model context-free grammars, they can nevertheless accurately represent complex linguistic phenomena and allow very efficient implementations.

A detailed formal introduction to cascading finite state transducers as well as examples for applications are given in the introduction of (Roche & Schabes, 1997).

Fidditch (Hindle, 1983) belongs to a category between finite-state parsing and context-free parsing. It uses context-free pattern-action rules, and a stack that is limited to three elements. It leaves most of the modifiers, adjuncts and relative clauses unattached. This parser was used for pre-processing the Penn Treebank annotations.

Another approach to chunking that uses a mixture of finite state and context-free techniques was presented by (Cardie & Pierce, 1998). They use NP rules of a pruned treebank grammar. During processing, at each point of a text is matched against the treebank rules and the longest match is chosen. This is related to our approach presented in section 4.6, where we also use context-free grammar rules induced from a treebank. The difference is that we use Markov Models for selection instead of the longest match and that our chunker can recognize the internal structure of chunks.

3.5 Markov Models and Weighted Finite-State Transducers

Recent investigations demonstrate that, in addition to Markov Models, weighted finite-state transducers are well suited for speech recognition and language processing tasks (e.g. Pereira & Riley, 1997; Tzoukermann & Radev, in press). The main difference in the representation is that Markov Models make outputs on states while transducers make outputs on transitions, which is the traditional difference between

Moore and Mealy machines. Another difference appears during processing. For a Markov Model, a sequence of outputs is given and the task is to search for the most-probable sequence of states that may have generated the output. For a transducer, a sequence of inputs is given, and the transducer creates a sequence of outputs. It is generally desirable that a transducer is sequential, i.e., the input side is deterministic. This is an advantage for processing because the time complexity is linear in the length of the sequence and independent of the number of states for sequential transducers. For Markov Models, it is also linear in the length of the sequence, but quadratic in the number of states.

However, processing Markov Models includes the "inversion" (input of the process is output of the model), and inverted transducers generally cannot be made sequential. Sequentialization is possible by generating two transducers, but handling the possibly enormous size of the resulting transducers is still a research topic. Additionally, several efficient training and smoothing techniques are already available for Markov Models.

On the other hand, transducers can be handled by finite state calculus. This was the motivation of Kempe (1997) Kempe to investigate the approximation of Markov Models with (unweighted) finite state transducers.

This thesis investigates sequential processing with Markov Models, which is currently one of the main techniques used for part-of-speech tagging. Additionally, it will be a very interesting topic for future research to exploit the presented type of sequential information with (weighted) transducers and to investigate if either Markov Models or transducers are better suited to natural language processing.

3.6 Automation of Corpus Annotation

Corpus resources are becoming increasingly more important, and the number of projects creating these resources is growing. Since the annotation of a corpus requires a lot of time-consuming, manual effort, automation of annotation is used to reduce this effort. The human annotator only creates part of the annotation, the rest is done by an automatic processing system. In older systems, the human annotator corrects the output of a parser that is used as a preprocessor. In newer systems, he directly interacts with some type of parser.

Annotation of the Penn Treebank was done with the help of Fidditch (Hindle, 1983) in a batch mode. The corpus was first processed by the parser. Its output was loaded into emacs and human correctors used a set of lisp functions to correct the parses based on a bracketed text representation. Annotation was reported to be very fast (750 – 1000 tokens per hour for trained annotators).

Current treebank projects switched to graphical representations. IceTree is the graphical annotation tool for the International Corpus of English (Greenbaum, 1996). Sentences are pre-processed by a parser specially designed for the project (the "survey parser") and subsequently corrected with the help of IceTree that offers a number of functions to manipulate trees and features that are associated with the nodes.

Part of the Czech National Corpus, the Prague Dependency Treebank, is annotated with dependency structures (Bemová et al., 1997; Hajic, 1998). They use a graphical tool that allows a number of operations on dependency structures. Annotation is done without a parser but with the help of interactive programs that automate the labeling.

The Treebanker (Carter, 1997) uses a different approach. This tool runs a parser in the background that creates a parse forest for the sentence currently annotated. The user sees a special graphical representation of parts that are ambiguous according to the grammar and selects or rejects nodes of the partial parse. All elements in the parse forest that are not compatible with the users decision are eliminated. This process proceeds until one parse remains. This procedure is very efficient in the reported project of annotating sentences in the ATIS (air travel information system) domain. But it requires a lexicon and parser that "license the correct analyses of utterances often enough for practical usefulness". Developing such a parser is a non-trivial task, especially if those corpora that should be built with the help of the tools are still missing.

In chapter 4, we will present a bootstrapping approach that does not require a full parser to start automation. Instead, it automizes easy tasks first and increases automation step by step, so that the amount of automation depends on the amount of training data that has been created previously. This type of automation together with a graphical user interface allowing all types of tree manipulations was used to create the NEGRA corpus (Skut et al., 1997) and the syntactically annotated parts of the Verbmobil corpora (Stegmann & Hinrichs, 1998).

Chapter 4

Tagging and Parsing with Markov Models

Chapter Summary

Markov Models as used in part-of-speech tagging are not restricted to the lowest level of a syntactic structure. We extend the technique, present models for assigning grammatical functions and phrase categories, and finally generate hierarchical structures with Cascaded Markov Models, where each layer is represented as a separate model. We present how to train these models on annotated corpora, yielding efficient and robust partial parsers.

4.1 Introduction

This chapter introduces new methods for syntactic language processing using Markov Models. We first improve and extend existing methods and then introduce a modification to the model in order to arrive at a partial parsing model.

As a starting point, we show that current tagging applications of Markov Models exploit only part of their powerful features. The majority of investigations on statistical part-of-speech tagging look at the best tag proposed by the tagger and ignore the ranking of alternatives and their actual probabilities. As we will see, information about alternatives and their probabilities can be a useful resource during corpus processing and corpus annotation.

Furthermore, part-of-speech taggers stick to the lowest levels in a syntactic structure: the words and their categories. But Markov Models can be applied at all levels of syntactic structure. We introduce the first statistical model for assigning grammatical functions that can handle a wide variety of user defined sets of grammatical functions. These are usually functions like *subject*, *object*, *modifier*, etc. Their number ranges from 12 in the English part of the Verbmobil corpus to 50 in the NEGRA corpus.

Then, Markov Models for tagging grammatical functions are further extended, and we introduce a robust model for the complete labeling task in a syntactic structure, covering part-of-speech tags (terminal nodes), grammatical functions (edges), and phrase categories (non-terminal nodes).

Markov Models are not restricted to the labeling task. We introduce a method of generating syntactic structures with their help. The output function of a Markov Model is modified so that the states are allowed to emit partial context-free structures instead of just single words or tags. The models take into account transitions from left to right within the structure. Using this sequential information is standard in part-of-speech tagging but it is new for non-sister nodes in context-free structures. We exploit the left-to-right transitional probabilities of terminal and non-terminal nodes, regardless of the hierarchical structure. Several Markov Models run in parallel, corresponding to the different layers in a syntactic structure. The result at a lower layer serves as input at the next higher layer. Note that the result of a layer corresponds to the states of a Markov Model, while the input of a layer corresponds to the output of the Markov Model.

Figure 4.1 shows how these tasks are encoded as Markov Models. The states of Markov Models represent part-of-speech tags and the outputs represent words when used for part-of-speech tagging. Moving up one layer, the states represent grammatical functions and the outputs represent tags (part-of-speech or phrase). Additionally, we need different Markov Models for different types of phrases because the distribution of labels varies with the type of phrase. For the next task, tagging phrase categories, states encode phrase categories *and* grammatical functions, so that they can be assigned simultaneously. And finally, the states represent tags (part-of-speech or phrase) and the output consists of words and partial structures for Cascaded Markov Models.

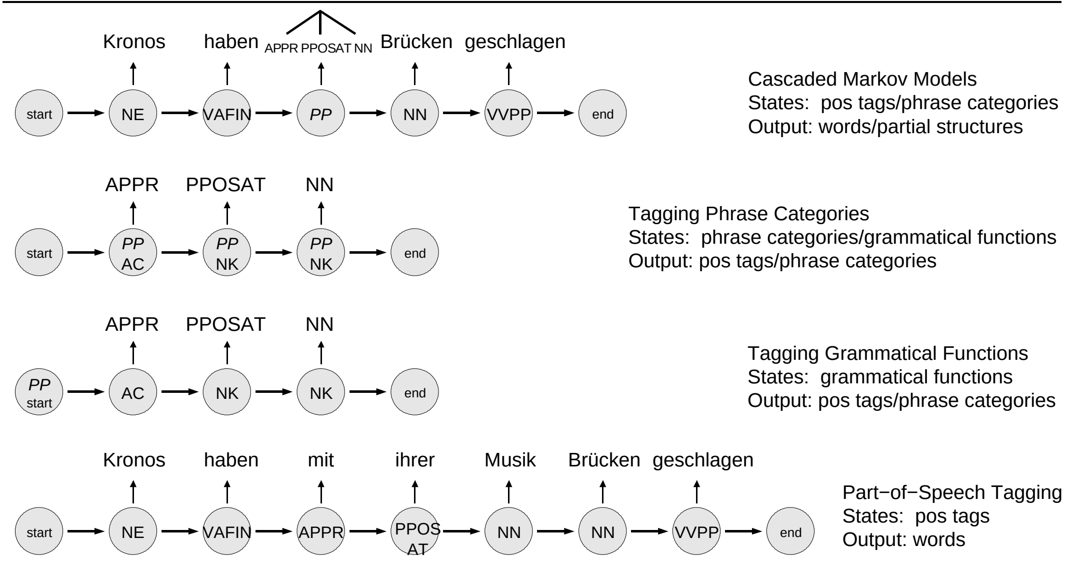

Kronos
haben
APPR PPOSAT NN
Brücken
geschlagen
start
NE
VAFIN
PP
NN
VVPP
end
Cascaded Markov Models
States: pos tags/phrase categories
Output: words/partial structures
APPR
PPOSAT
NN
PP AC
PP NK
PP NK
Tagging Phrase Categories
States: phrase categories/grammatical functions
Output: pos tags/phrase categories
PP start
AC
NK
Tagging Grammatical Functions
States: grammatical functions
Output: pos tags/phrase categories
mit
ihrer
Musik
APPR
PPOSAT
Part−of−Speech Tagging
States: pos tags
Output: words

Figure 4.1: Markov Models at different processing levels. Starting with part-of-speech tagging (bottom), the technique is extended to tagging grammatical functions, tagging phrase categories and finally Cascaded Markov Models that generate syntactic structures.

4.2 Part-of-Speech Tagging

Statistical part-of-speech tagging is usually performed by efficient techniques that already ensure high tagging accuracy:

- trigram models (second order Markov Models);
- Viterbi algorithm for efficient processing (e.g. Rabiner, 1989);
- handling sparse data by linear interpolation (Brown et al., 1992), maximum entropy models (Ratnaparkhi, 1996), successive abstraction (Samuelsson, 1996), Good-Turing estimation (Good, 1953), Katz-Backoff (Katz, 1987), and others;
- handling unknown words with a suffix trie (Samuelsson, 1993).

Nevertheless, there are some drawbacks during processing which need to be eliminated in order to further increase tagging accuracy.

The first step is to carefully clean the training and test corpora. A large number of errors stem from inconsistencies in manual annotations. This has already been observed in (Karlsson et al., 1994) and (Bod, 1995). The former designed their corpus and tagset with respect to intra- and inter-annotator consistency. They avoided categories that are hard to distinguish and intentionally left ambiguity in their tagsets. The latter performed an additional manual cleaning step before training and testing. Both groups improved tagging and parsing results by this additional effort.

Our investigations confirm this effect. The Stuttgart-Tübingen tagset contains some classes that are difficult to annotate consistently. For example, it is sometimes unclear whether to use ADJD (adjective, used predicatively) and ADV (adverb):

die	*Menge*	*ist*	*endlich*(ADJD)	vs.	*er*	*kommt*	*endlich*(ADV)
the	*set*	*is*	*finite*		*he*	*comes*	*finally*

These tags are frequently confused even in corrected texts.

Another example is the confusion of PIAT (attributive indefinite pronoun) and PIDAT (attributive indefinite pronoun with determiner):

zuviele(PIAT)	*Fragen*	vs.	*beide*(PIDAT)	*Fragen*
too many	*questions*		*both*	*questions*

Also very difficult is the distinction of common nouns (NN) and proper nouns (NE), e.g., *Mozartstraße* is classified as NN, and *Bodensee* is classified as NE according to the Stuttgart-Tübingen-Tagset. As far as closed-class words are concerned, this type of errors can be cleaned up using a program that tests the annotation against a fixed list of allowed tags. This additional processing step improves tagging accuracy in the NEGRA corpus by up to 0.5%.

The techniques used for tagging can be improved. One way of increasing the lexical coverage of the tagger (i.e., a token found in new text is less often "out of vocabulary") is to use an external, manually created lexicon or morphological component. Schneider and Volk (1998) reported an improvement of around 0.8% in tagging accuracy when using GertWol (Haapalainen & Majorin, 1995) for analyzing words that are unknown to the tagger.

If neither a lexicon nor a morphological component is available, a large, untagged corpus can be used to estimate lexical parameters in the case of unknown words. A tagged corpus is used to train a model, which in turn is used to tag a large, untagged corpus. Now, lexical frequencies are generated from such a corpus for all words that are not seen in the annotated part. Using lexical frequencies for unknown words generated from 40 million tokens of untagged text increased accuracy for the NEGRA corpus by about 0.3%.

We do not use a re-estimation of the complete model (Baum-Welch re-estimation) using the large, untagged corpus because it does not necessarily improve the model when it is already trained on a sufficient portion of annotated text. This was presented by Elworthy (1994) and we can confirm his results. Accuracy decreased when re-estimating on untagged data after training on a large tagged corpus.

Yet another possibility to improve the accuracy of a part-of-speech tagger is to add hand-crafted filters for post-processing tagger output. A frequent error when using the Stuttgart-Tübingen tagset is the confusion of finite and non-finite verbs. These account for around 15% of the errors (0.6% of all tags):

...,	*weil*	*mehr*	*Kinder*	*in*	*die*	*Konzerte*	*kommen/*VVFIN	.
(...,	*because*	*more*	*children*	*to*	*the*	*concerts*	*come*	*.)*

The tagger erroneously tags *kommen* in this position as non-finite verb (VVINF). When processing the word, the beginning of the clause is out of the tagging window. Therefore, the tagger does not know if it should process a verb final clause starting with a complementizer or a verb second clause with a finite verb at the second position. Disambiguation is performed mainly on local context, which is not sufficient for this case. A manually added finite-state filter that detects finite verbs and/or complementizers at the beginning of clauses can correct several of these errors.

4.3 Alternatives and Reliability

An advantage of statistical models over non-statistical models is their ability to explicitly rank alternatives. This advantage is only partially exploited in standard tagging methods. They always choose the alternative that has the highest probability. But the assignments that are ranked second, third, etc. also contain important information. In certain cases, it is the second, third, ...-best alternative that represents the correct assignment, especially if their probabilities are close to that of the best alternative. There should be a difference in a second rank having a probability very close to the "winner" and a second rank that has a much smaller probability.

The following investigates alternative solutions when assigning tags, but the methods can also be applied when structures are to be determined.

Calculating alternatives can serve two purposes: estimating the reliability of assignments and keeping ambiguity in the output. Both are investigated in the following sections.

4.3.1 Reliability

How good is the best? Alternative solutions can give an answer to this question. Intuitively, the best should be the better the further below other solutions are. Let k be the number of possible states q_i of a Markov Model for a word, and let q_1 be the best state according to some probability model. Then, the probability of the best state is $P(q_1)$. The probability that q_1 is *not* the best state according to the model is

$$P(\textbf{not } q_1) = 1 - P(q_1) = \sum_{i=2}^{k} P(q_i) \tag{4.1}$$

The probability of a specific state q_i at position t is the sum over all probabilities of sequences with a given length T traversing the state at the particular position:

$$P(q_i) = \sum_{q_{i_1},\ldots,q_{i_{t-1}},q_i,q_{i_{t+1}},q_{i_T} \in Q^T} P(q_{i_1},\ldots,q_{i_{t-1}},q_i,q_{i_{t+1}},q_{i_T}) \tag{4.2}$$

The sum cannot be calculated in a naive way, since the number of possible sequences grows exponentially in the length of the sequence. Therefore, we use a dynamic programming algorithm.

The Forward-Backward-Algorithm (Baum et al., 1970) defines two accumulators $\alpha_t(i)$ and $\beta_t(i)$ for time t and state i. $\alpha_t(i)$ is defined as the forward probability, i.e., the probability of generating the partial output sequence $w_1 \ldots w_t$ and being in state q at time t (given some model M). $\beta_t(i)$ is defined as the backward probability, i.e., starting at the end of the sequence and generating the partial output sequence $w_{t+1} \ldots w_T$ and being in state q_i at time t (see section 2.2.3 for the formula definitions).

We are looking for a third value, the probability of generating the complete sequence $w_1 \dots w_T$ and traversing state q_i at time t. These are the gamma probabilities, which can be expressed by the forward and backward accumulators:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)} \tag{4.3}$$

Since the α and β accumulators can be calculated in linear time $O(T)$, this is also true for the γ probabilities.

We expect the tagging accuracy at time t to be the higher the larger the γ value of the best alternative $\gamma_t(best)$. Since the γ probabilities sum up to one,

$$\sum_i \gamma_t(i) = \sum_i \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)} = 1 \tag{4.4}$$

we can empirically test this hypothesis by measuring tagging accuracies depending on absolute values $\theta_\gamma \in [0..1]$, i.e., the tagging accuracy if

$$\gamma_t(best) > \theta_\gamma \quad \text{vs.} \quad \gamma_t(best) \le \theta_\gamma. \tag{4.5}$$

Probabilities within a sequence of states are combined multiplicatively. Therefore, we will prefer a relative measure using the quotient of two probabilities instead of an absolute value. This allows the ommission of normalization, which is usually used for taggers to increase tagging speed. So we will look at tagging accuracies if

$$\frac{\gamma_t(best)}{\gamma_t(\mathbf{not}\ best)} = \frac{\gamma_t(best)}{\sum_{i \ne best} \gamma_t(i)} = \frac{\alpha(best)\beta(best)}{\sum_{i \ne best} \alpha(i)\beta(i)} > \theta \tag{4.6}$$

vs.

$$\frac{\alpha(best)\beta(best)}{\sum_{i \ne best} \alpha(i)\beta(i)} \le \theta \tag{4.7}$$

Thresholds on the absolute value of γ and on the quotient can be converted into each other, having

$$\theta = \frac{\theta_\gamma}{1 - \theta\gamma} \tag{4.8}$$

Additionally, using relative values allows to use the Viterbi approximation in the denominator, replacing the sum by the maximum, which, in this case, are the values for the alternative second best state:

$$\frac{\alpha(best)\beta(best)}{\alpha(alt)\beta(alt)} > \theta \quad vs. \quad \frac{\alpha(best)\beta(best)}{\alpha(alt)\beta(alt)} \le \theta \tag{4.9}$$

This further reduces the number of calculations. For a convenient notation, we will use the equivalent quotient $\gamma_t(best)/\gamma_t(alt)$, but for computation, we will use (4.9).

4.3.2 Remaining ambiguity

Informally, the principle is "not to decide when uncertain". So we leave selected ambiguity in the output. The criterion for keeping a tag is the quotient (4.9).

We select a threshold $\theta \geq 1$. For each position t, we calculate the best state q_{best} and additionally keep all states q_{alt} having

$$\gamma_t(alt) \geq \frac{\gamma_t(best)}{\theta}. \tag{4.10}$$

Using this mechanism, a tagger generates possibly ambiguous output. An alternative to ambiguous (multi-tag) output is to combine tags that are frequently confused and always emit one inherently ambiguous tag. As an example, common nouns and proper nouns are usually difficult to distinguish for a part-of-speech tagger. For a tagger based on the Stuttgart-Tübingen tagset, almost half of the errors stem from a confusion of these two tags (NN for common nouns, NE for proper nouns). Thus, a solution could be to remove NN and NE from the output and instead use a combined tag (e.g., NE-NN). This would increase accuracy by more than 1%[1], which is very tempting at the first sight, but there is a big advantage of ambiguous (multi-tag) output. The difference is indicated by the example in figure 4.2.

The leftmost column contains a text ("*a rich farmer in Sweden*"). Both *Bauer* (*farmer* or a surname) and *Schweden* (the country *Sweden* or its inhabitants) are ambiguous and can be common noun (NN) or proper nouns (NE). The second column shows the possible tags according to the lexicon. The third column represents the output of a tagger that always combines NN and NE, and the fourth column represents the output of a tagger that emits multiple tags together with their probabilities. All tags having probabilities that are more than 100 times smaller than the best assignment are pruned ($\theta = 100$) and the remaining probabilities are normalized.

[1]This is a possibitiy of making the problem simpler in order to achieve higher accuracy rates. See also (Brants, 1995)

(1) Text	(2) Lexicon	(3) combined noun tag	(4) multi-tag output (tag and probability)			
Ein	ART CARD PTKVZ	ART	ART	1.00		
reicher	ADJA	ADJA	ADJA	1.00		
Bauer	NE NN	NE-NN	NN	0.92	NE	0.08
in	APPR	APPR	APPR	1.00		
Schweden	NE NN	NE-NN	NE	0.94	NN	0.06

Figure 4.2: Difference of a tagger that emits combined tags (3) and a tagger that emits multiple tags and their probabilities (4)

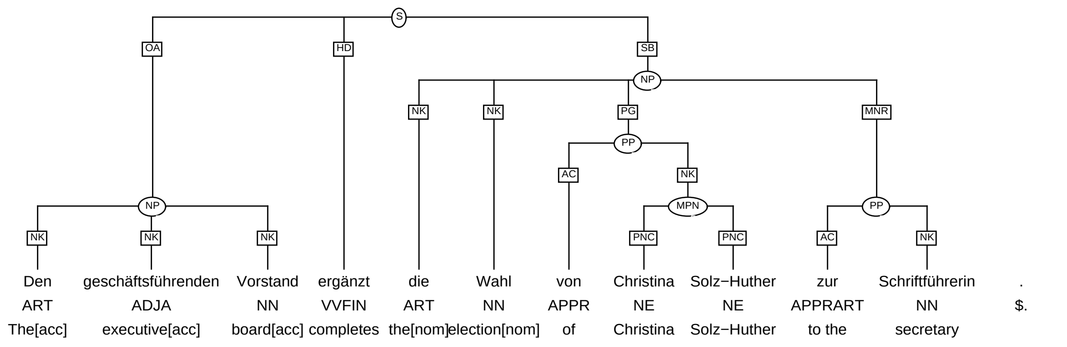
The output in column (4) is the one that is most informative. First, all cases that can be safely decided on the context will get a unique tag, which is an advantage in further processing steps. Second, in cases that cannot be decided safely, probabilities are assigned to the alternatives, and again this is usually a big advantage in further processing.

In the example, probabilities strongly suggest one of the readings, NN for *Bauer* and NE for *Schweden*, which are the correct readings. The opposite assignments would result in a corrupt meaning. The decision which readings should be left in the analysis, can be based on the probabilities calculated by the tagger. The threshold θ was set to 100 in this example, which resulted in the removal of the CARDinal and separable verb prefix (PTKVZ) readings for *Ein*.

4.4 Tagging Grammatical Functions

The relation of a constituent to its immediately dominating phrase is expressed as a *grammatical function*. Each constituent of a phrase can be assigned a particular function, e.g., an NP within an S node can be the *subject*, an *object*, some *adjunct*, etc. For some of these functions, there is a chance of identifying them by the category of the child node, e.g., in most theories, the finite verb under an S node is the *head* of the sentence. Other elements may be identified by their order, e.g., the NP just in front of a finite verb is in most cases the subject in English sentences.

In a general approach, all these functions are explicitly labeled. Figure 4.3 shows



'The election of Christina Solz-Huther as secretary completes the executive board'

Figure 4.3: Example sentence. The structure consists of terminal nodes (words and their parts-of-speech), non-terminal nodes (phrases) and edges that are labeled with grammatical functions.

an example annotation. There, grammatical functions are associated to the edges between the affected nodes. The sentence is taken from the NEGRA corpus (Skut et al., 1997). The Penn Treebank also contains information about grammatical functions, but only for a small fraction of the relations. The function is associated with the child node, separated from the child's category by a dash (cf. figure 4.4).

This section reports on a statistical approach to learning grammatical functions and assigning them to previously unseen data. To our knowledge, this is the first general approach to take account of all classes of grammatical functions on a statistical basis. The only comparable approach was presented by Lima (1997) who makes a decision between subject and direct object. A description of her approach is given in section 3.2.

4.4.1 The Method

The basic idea of the approach is to use standard part-of-speech tagging techniques at the next higher level in the syntactic structure. A part-of-speech tagger assigns part-of-speech tags to words. The presented tagger for grammatical functions assigns functions to part-of-speech tags and phrase categories. Additionally, we expect the functions of constituents to be different if they appear in phrases of different types. So, instead of using exactly one distribution for all contextual and lexical probabil-

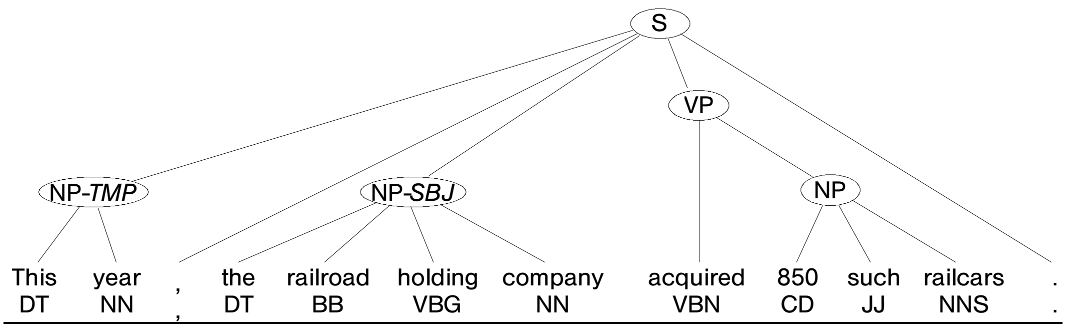


Figure 4.4: Example sentence from the Penn Treebank. Grammatical functions are associated to the child node of a phrase and are represented together with the child's category, separated by a dash.

ities $P(\cdot)$, the tagger for grammatical functions works with lexical and contextual probability measures $P_Q(\cdot)$ depending on the category of the mother node (Q). Each phrase category (S, VP, NP, PP etc.) is represented by a different Markov Model. The categories of the daughter nodes correspond to the outputs of the Markov Model, while grammatical functions correspond to states.

The structure of a sample sentence is given in figure 4.3. Figure 4.5 shows those parts of the Markov Models for sentences (S) and verb phrases (VP) that represent the correct paths for the example.[2]

Given a sequence of word and phrase categories $T = T_1 \dots T_k$ and a parent category Q, we calculate the sequence of grammatical functions $G = G_1 \dots G_k$ that link T and Q as

$$\underset{G}{\operatorname{argmax}} P_Q(G|T) \tag{4.11}$$

$$= \underset{G}{\operatorname{argmax}} \frac{P_Q(G) \cdot P_Q(T|G)}{P_Q(T)}$$
$$= \underset{G}{\operatorname{argmax}} P_Q(G) \cdot P_Q(T|G)$$

Assuming the Markov property we have

[2]cf. appendix A for a description of tags used in the example

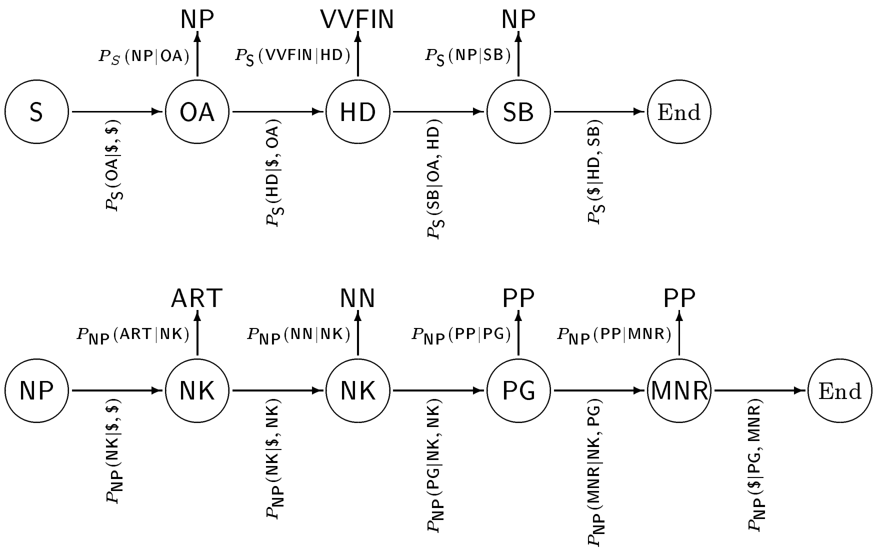


Figure 4.5: Parts of the Markov Models used to generate grammatical functions for the S node and the right NP node in the sentence of figure 4.3. All unused states, transitions and outputs are omitted. Models for the other nodes are built analogously.

$$P_Q(T|G) = \prod_{i=1}^{k} P_Q(T_i|G_i) \tag{4.12}$$

and

$$P_Q(G) = \prod_{i=1}^{k} P_Q(G_i|C_i) \tag{4.13}$$

The contexts C_i are modeled by a fixed number of surrounding elements. We take into account a window of two contextual grammatical functions, which results in a trigram model:

$$P_Q(G) = \prod_{i=1}^{k} P_Q(G_i|G_{i-2}, G_{i-1}) \tag{4.14}$$

The same types of smoothing as in standard part-of-speech tagging can be applied to the technique of tagging grammatical functions, e.g., linear interpolation of unigrams, bigrams, and trigrams (Brown et al., 1992), successive abstraction (Samuelsson, 1996), Good-Turing estimation (Good, 1953), Katz-Backoff (Katz, 1987), and others. And the same types of training can be used, i.e., relative frequencies from an annotated corpus or the Baum-Welch algorithm (Baum et al., 1970).

4.4.2 Encoding of States in the Markov Model

Several investigations have shown that the accuracy of taggers and parsers depends heavily on the encoding of categories (e.g. Elworthy, 1995; Johnson, 1998; Skut & Brants, 1998). We confirm these results for the assignment of grammatical functions.

A simple transfer of the part-of-speech tagging technique encodes grammatical functions as states, and tags (part-of-speech and phrase) as outputs of a Markov Model (cf. figure 4.5). This already yields good results, but it can be improved on.

An alternative approach encodes both elements, the grammatical function and the part-of-speech or phrase tag, in the state, such that the transitions not only depend on the previous function but also on the previous tag. This encoding represents a type of "lexicalization", in analogy to part-of-speech tagging, where lexicalization means to encode word information in the states in addition to tags. For grammatical function tagging, which is one level higher than part-of-speech tagging, it means to encode category information in addition to the functions.

We encode a function tag as a feature structure with two elements:

$$\begin{bmatrix} \text{CAT:} & T \\ \text{FUN:} & G \end{bmatrix}$$

with syntactic category T of the corresponding child node and grammatical function G. So instead of encoding the function *subject* as SB, we use

$$\begin{bmatrix} \text{CAT:} & \mathsf{NN} \\ \text{FUN:} & \mathsf{SB} \end{bmatrix} \begin{bmatrix} \text{CAT:} & \mathsf{NP} \\ \text{FUN:} & \mathsf{SB} \end{bmatrix} \begin{bmatrix} \text{CAT:} & \mathsf{S} \\ \text{FUN:} & \mathsf{SB} \end{bmatrix} \dots$$

for a single common noun being the subject, for a subject NP, or for a sentence being the subject, etc.

This encoding simplifies the calculation of the maximization formula, since the lexical probabilities are reduced to

$$P_Q(T_i|G_i) = \begin{cases} 1 & \text{if } G_i = \begin{bmatrix} \text{CAT:} & T_i \\ \text{FUN:} & X \end{bmatrix} \text{ for some gram. function } X \\ 0 & else \end{cases} \tag{4.15}$$

As we will see in the chapter on evaluation (7), this change in encoding significantly improves tagging accuracy (around 1% in the NEGRA corpus).

The Markov Model implicitly models the beginning of the phrase because of the term $P_Q(G_1|G_{-1}, G_0)$ in equation 4.14. G_{-1} and G_0 are not part of the sequence that is to be tagged, but are defined as special "start of sequence" tags. The end of a phrase is not modeled, the product ends at the last tag G_k. The formula can be extended and the product runs up to $k+2$. G_{k+1} and G_{k+2} are defined as special "end of sequence" tags, so that the phrase has anchors at both the start and the end. Two additional elements are necessary to account for the trigram model. Explicitly modeling the end of a phrase improves results slightly (improvement in the NEGRA corpus: 0.2%).

4.4.3 Morphological Information

Very often, grammatical functions coincide with morphological (inflectional) features. As examples for German, the function *subject* coincides with nominative case, the function *direct object* coincides with accusative case, and the function *head verb* of a sentence is assigned to a finite verb. Knowledge about these features should help in tagging grammatical functions when processing inflected languages.

Our method of assigning grammatical functions is based on part-of-speech tags. Unfortunately, the representation of inflection in a syntactic tagset is usually very coarse-grained. There is a good reason for leaving out morphological information in the tagset: morphological analyses without context are usually highly ambiguous, and local context as used by part-of-speech taggers is not sufficient for resolving these ambiguities. Elworthy (1995) showed that tagging accuracy significantly decreases when assigning information about case, gender and number. Hajic and Hladka (1998) introduce additional techniques in order to handle morphological information in a tagset.

We confirmed these findings on the Stuttgart-Tübingen tagset (STTS) for German. The tagset distinguishes four different forms of verbs (infinite, finite, past participle, and imperative), but is uninformative about gender, number, and case of adjectives and nouns. There is an extended version of the tagset encoding morphological information, but it is impractical to use it for part-of-speech tagging. Just adding case information causes accuracy to drop to around 85%, which is a very low result for a part-of-speech tagger.

But for tagging grammatical functions, morphology is exactly the type of information that is lacking. Looking at the list of errors in section 7.3 reveals that very often the functions *subject* and *direct object* are confused, or that the tagger cannot distinguish *postnominal genitives* and *appositions*. In both cases, case information about the NPs should solve the problem.

Given the fact that a reliable unique morphological analysis cannot be assigned by part-of-speech tagging, our approach exploits information delivered by an ambiguous (underspecified) morphological analysis.

A morphological analyzer can assign the set of possible analyses with high accuracy. So we know that a word can have some morphological features, but definitely does not have some others (e.g., *die Männer* can be nominative or accusative plural, but not genitive or dative and not singular).

We encode morphological information as features together with category information. Looking only at case information, an NP that is ambiguous for nominative and accusative is encoded as

$$\begin{bmatrix} \text{CAT:} & \mathsf{NP} \\ \text{CASE:} & \{\mathsf{nom}, \mathsf{acc}\} \\ \text{FUN:} & X \end{bmatrix}$$

This means that the NP can be nominative or accusative, but not genitive or dative. The ambiguous assignment is made with very high accuracy, as opposed to the disambiguated tags:

$$\begin{bmatrix} \text{CAT:} & \mathsf{NP} \\ \text{CASE:} & \{\mathsf{nom}\} \\ \text{FUN:} & X \end{bmatrix} \quad \begin{bmatrix} \text{CAT:} & \mathsf{NP} \\ \text{CASE:} & \{\mathsf{acc}\} \\ \text{FUN:} & X \end{bmatrix}$$

The type of information added to the tagset needs to be carefully selected in order not to run into sparse data problems.

Adding information to tags for particular words is, e.g., used in the CASS partial parser (Abney, 1996). There, tags for particular words are "fixed" before applying the parser; hence, this technique was named *tagfixing*. For tagging grammatical functions, tags are fixed based on the words' morphology. Results improved when using tagfixes based on word identities. We will show that tagfixes based on morphology also improve accuracy significantly. Adding the morphological analyzer Morphix (Finkler & Neumann, 1988) when processing the NEGRA corpus improves accuracy of grammatical functions by 0.5 – 1.0% (cf. section 7.3).

4.5 Assigning Phrase Categories

Part-of-speech tagging assigns syntactic categories to words. Tagging grammatical functions assigns functions to syntactic categories that are dominated by a phrase node. This section introduces a tagger for phrase categories and thereby completes the labeling of a syntactic structure. The presented technique is an extension of assigning grammatical functions.

The task can be described by using a context-free rule as an example: given the right-hand side of a rule $\mathsf{X} \rightarrow \alpha$; find the phrase label for X that has the highest probability. There is more than one candidate in a large number of cases. As an example, the sequence NP ADJP PP occurs in the Penn Treebank (Wall Street Journal part) 29 times as an NP and 18 times as an S, e.g.,

NP → [$_{\mathsf{NP}}$ *financing*] [$_{\mathsf{ADJP}}$ *as low as 6.9 %*] [$_{\mathsf{PP}}$ *on 24-month loans*]

S → [$_{\mathsf{NP}}$ *short skirts*] [$_{\mathsf{ADJP}}$ *not welcome*] [$_{\mathsf{PP}}$ *in Texas court*]

Additionally, sequences that did not occur in the training data may form a phrase in new text, so robust processing requires the assignment of a category to unknown sequences.

4.5.1 The Method

The method of assigning grammatical functions as presented in the previous section can be extended to recognize phrase categories. There, different Markov Models for each category were introduced, and the phrase category was known before assigning grammatical functions using the appropriately chosen model.

In order to assign the phrase label automatically, we run all models in parallel. Each model assigns grammatical functions and, more important for this step, a probability to the phrase. The model assigning the highest probability is assumed to be the most adequate, and the corresponding label is assigned to the phrase.

Formally, we calculate the phrase category Q by summing over all sequences of grammatical functions $G = G_1 \ldots G_k$ on the basis of the sequence of daughters $T = T_1 \ldots T_k$ with

$$\operatorname*{argmax}_{Q} \sum_{G} P_Q(G|T). \tag{4.16}$$

The complexity of summing over G is linear in the length of the sequence when using a Markov Model (n-grams) and dynamic programming: $O(|\mathcal{T}|^n|G|)$. We need to calculate the sum for each type of phrase separately, so this involves an additional constant factor $|\mathcal{Q}|$, resulting in $O(|\mathcal{T}|^n|G||\mathcal{Q}|)$.

Assuming that the sum in 4.16 is composed of one large element and a large number of small elements that change the result only marginally, we can replace the formula by its *Viterbi approximation* which uses maximization instead of summation:

$$\operatorname*{argmax}_{Q} \max_{G} P_Q(G|T). \tag{4.17}$$

Calculations are simplified when using maximization because we can calculate the maximum and the actual sequence of grammatical functions G at the same time. This procedure is equivalent to a different view on the same problem involving one large (combined) Markov Model that enables a very efficient calculation of the maximum.

Let $\mathcal{G}_Q$ be the set of all grammatical functions that can occur within a phrase of type Q. Assume that these sets for different phrases are pairwise disjoint. One can achieve this property by indexing all used grammatical functions with their associated phrases and, if necessary, duplicating labels, e.g., instead of using *head*, *modifier*, ..., use the indexed labels *head of* S, *head of* VP, *modifier of* NP, ... This property makes it possible to determine a phrase category by inspecting the grammatical functions involved.

The union of the Markov Models can be produced by introducing a new start state with transitions to the original start states. The probabilities of the new transitions are set according to the a-priori probabilities of the corresponding phrase types.

When applied, the combined model assigns grammatical functions to the elements of a phrase (not knowing its category in advance). If transitions between states representing labels with different indices are forced to zero probability (smoothing is applied to all other transitions), all labels that are assigned to a phrase have the same index. This uniquely identifies a phrase category.

The two additional conditions

$$G \in \mathcal{G}_{Q1} \Rightarrow G \notin \mathcal{G}_{Q2} \quad (Q_1 \neq Q_2) \tag{4.18}$$

and

$$G_1 \in \mathcal{G}_Q \wedge G_2 \notin \mathcal{G}_Q \Rightarrow P(G_2|G_1) = 0 \tag{4.19}$$

are sufficient to calculate

$$\operatorname*{argmax}_G P(G|T) \tag{4.20}$$

using the Viterbi algorithm and to identify both the phrase category and the respective grammatical functions.

4.5.2 Encoding of States

Encoding of information in states of the Markov Model and the type of information that is selected affect the accuracy of the model. Results of different encodings for tagging grammatical functions are also valid for assigning phrase categories. The tagging accuracy is influenced by applying tagfixes, but as the evaluation shows, the influence is smaller for phrase labels than for grammatical functions.

Additionally, a different encoding of states can eliminate the difference between summation and maximization. If the states no longer represent syntactic categories (of the child nodes) and grammatical functions (the relation between a child node and its parent), but only the syntactic category, we give up the advantage of calculating the phrase label and grammatical functions simultaneously. The resulting model, however, has either exactly zero or exactly one path for each phrase category and a given sequence of the children's categories. Thereby, summation and maximization result in the same probabilities.

This reduced encoding is isomorphic to the encoding presented in the previous section together with summation, which demonstrates the big influence that the type of encoding can have.

4.6 Cascaded Markov Models

The previous sections addressed the labeling problem, i.e., assigning part-of-speech tags, grammatical functions, and phrase categories. We discuss now the generation of hierarchical structures with the help of Markov Models.

The basic idea is to construct the parse layer by layer, first structures of depth one, then structures of depth two, and so forth (cf. figure 4.6). For each layer, a Markov Model determines the best set of phrases. These phrases are used as input for the next layer, which adds one more layer. Phrase hypotheses at each layer are generated by stochastic context-free rules and filtered from left to right by Markov Models.

4.6.1 Tagging Lattices

When encoding a part-of-speech tagger as a Markov Model, states represent syntactic categories[3] and outputs represent words. Contextual probabilities of tags are encoded as transition probabilities of tags, and lexical probabilities are encoded as output probabilities of words in states.

We introduce a modification to this encoding for parsing. States additionally may represent nonterminal categories (phrases). These new states emit partial parse trees (cf. figure 4.7). This can be seen as collapsing a sequence of terminals into one non-terminal. Transitions into and out of the new states are performed in the same way as for words and parts-of-speech.

We use stochastic context-free grammar rules, learned from a corpus, to create phrase hypotheses at each layer. All rules with right sides that are compatible with part of the word sequence are added to the search space. Figure 4.8 shows an example for hypotheses at the first layer. Each bar represents one hypotheses.

[3]Categories and states directly correspond in bigram models. For higher order models, tuples of categories are combined to one state.

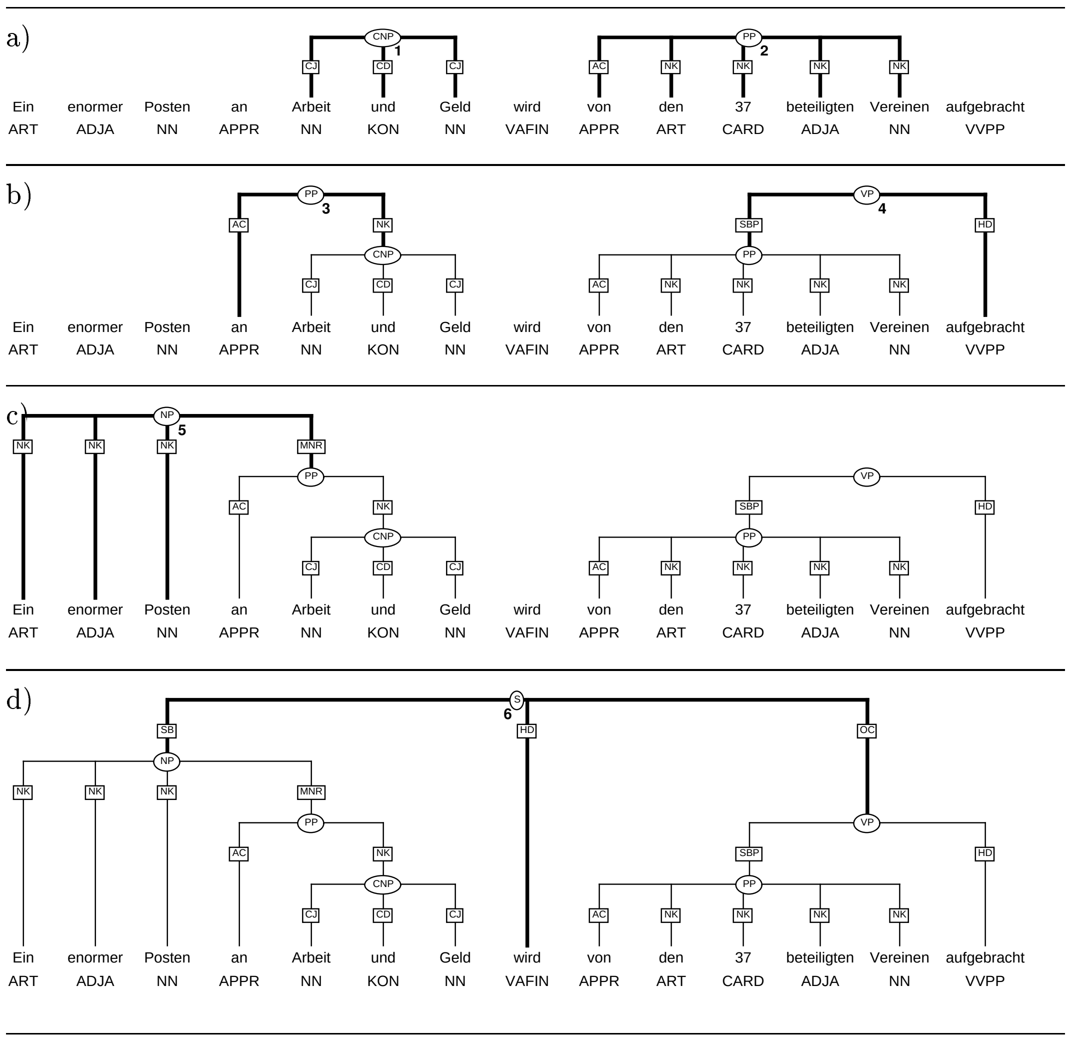

a)
b)
c)
d)
CNP
PP
VP
NP
S
1
2
3
4
5
6
CJ CD CJ
AC NK NK NK NK
SBP HD
MNR
SB HD OC
Ein enormer Posten an Arbeit und Geld wird von den 37 beteiligten Vereinen aufgebracht
ART ADJA NN APPR NN KON NN VAFIN APPR ART CARD ADJA NN VVPP

Figure 4.6: Creating a structure layer by layer (and from left to right). The creation order is indicated by indices.

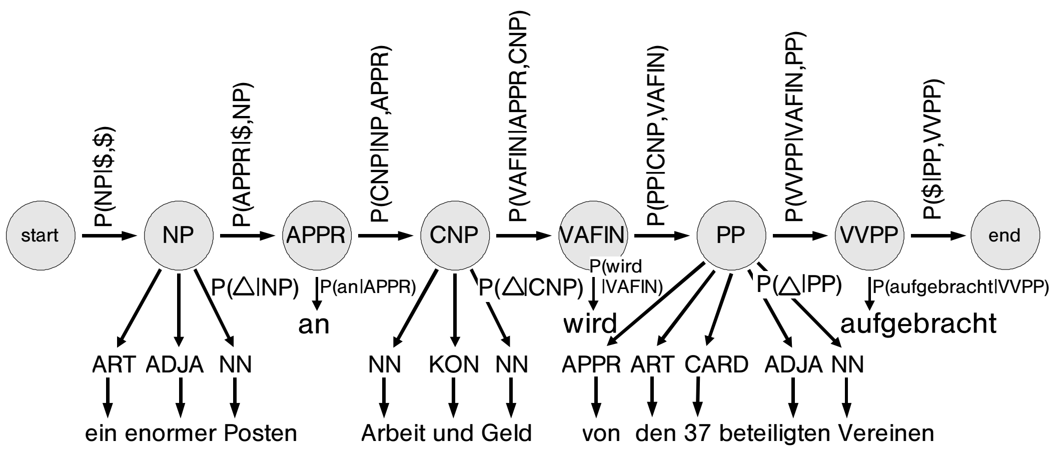


Figure 4.7: Part of the Markov Model used to generate the best path in figure 4.8. Opposed to part-of-speech tagging, outputs can consist of structures with probabilities according to a stochastic context-free grammar.

The position of the bar indicates the covered words. It is labeled with the type of the hypothetical phrase, an index in the left upper corner for later reference, the negative logarithm of this phrase generating the particular words (i.e., the smaller the better; probabilities for part-of-speech tags are omitted for clearness). This part is very similar to chart entries of a chart parser.

All phrases that are newly introduced at this layer are marked with an asterisk (*). They are produced according to context-free rules based on the elements passed from the next lower layer. The layer below layer 1 is the part-of-speech layer.

As an example, edge #15 represents an NP. It is generated because the context-free grammar contains the rule NP → ART ADJA. It covers the words from position 0 to 2, and its probability of generating the terminals is

$$P(\mathsf{NP} \to \mathsf{ART\ ADJA}, \mathsf{ART} \to \text{ein}, \mathsf{ADJA} \to \text{enormer}) = 10^{-6.60}.$$

The probability is estimated by the probability of the context-free derivation, thus it is the product of the probabilities of all context-free rules that are involved. They are taken from a stochastic context-free grammar (these probabilities are not printed

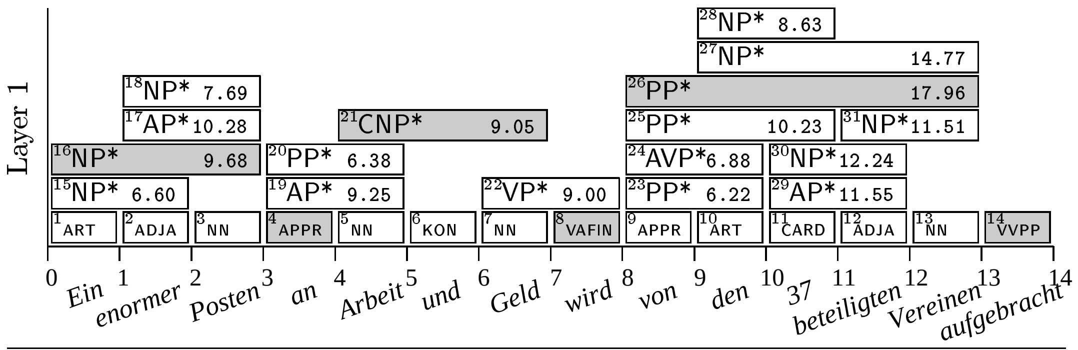

Layer 1
28NP* 8.63
27NP* 14.77
18NP* 7.69
26PP* 17.96
17AP* 10.28
21CNP* 9.05
25PP* 10.23
31NP* 11.51
16NP* 9.68
20PP* 6.38
24AVP* 6.88
30NP* 12.24
15NP* 6.60
19AP* 9.25
22VP* 9.00
23PP* 6.22
29AP* 11.55
1ART 2ADJA 3NN 4APPR 5NN 6KON 7NN 8VAFIN 9APPR 10ART 11CARD 12ADJA 13NN 14VVPP
0 Ein 1 enormer 2 Posten 3 an 4 Arbeit 5 und 6 Geld 7 wird 8 von 9 den 10 37 11 beteiligten 12 Vereinen 13 aufgebracht 14

Figure 4.8: Phrase hypotheses according to a context-free grammar for the first layer. Hypotheses marked with an asterisk (*) are newly generated at this layer, the others are passed from the next lower layer (layer 0: part-of-speech tagging). Numbers to the right of the phrase symbols indicate negative logarithms of probabilities of the respective context-free sub-trees. The best path according to a Markov Model trained on the NEGRA corpus is marked grey.

in the figure):

$$P(\mathsf{NP} \rightarrow \mathsf{ART\ ADJA}) = 10^{-2.30};$$

and from the part-of-speech tagger:

$$\begin{aligned} P(\mathsf{ART} \rightarrow \text{ein}) &= 10^{-1.13}, \\ P(\mathsf{ADJA} \rightarrow \text{enormer}) &= 10^{-3.17}; \end{aligned}$$

yielding

$$P(\mathsf{NP} \rightarrow \mathsf{ART\ ADJA}) \cdot P(\mathsf{ART} \rightarrow \text{ein}) \cdot P(\mathsf{ADJA} \rightarrow \text{enormer}) = 10^{-6.60}.$$

The hypotheses form a lattice, with the word boundaries being states and the phrases being edges. Selecting the best hypotheses means to find the best path from node 0 to the last node (node 14 in the example). The best path can be efficiently found with the Viterbi algorithm, which runs in time linear to the length of the word sequence. Processing of a layer is similar to word lattice processing in speech recognition (see e.g. Samuelsson, 1997).

But we do not only want to take into account edge probabilities, representing the context-free partial-parse, but also contextual probabilities. Therefore, two types of probabilities are important when searching for the best path in a lattice. The first are those probabilities that are already indicated in figure 4.8. These are probabilities of the hypotheses (phrases) producing the underlying terminal nodes (words). The second type are context probabilities, i.e., that some type of phrase follows or precedes another. We choose a Markov Model for representing the latter type of probabilities. The two types of probabilities coincide with lexical and contextual probabilities of a Markov Model, respectively.

According to a trigram model created from the NEGRA corpus, the path in figure 4.8 that is marked grey is the best path in the lattice. Its probability is composed of

$$\begin{aligned} P_{best} = {} & P(\mathsf{NP}|\$,\$)P(\mathsf{NP} \Rightarrow^* \textit{ein enormer Posten}) \\ & \cdot P(\mathsf{APPR}|\$,\mathsf{NP})P(\mathsf{APPR} \rightarrow \textit{an}) \\ & \cdot P(\mathsf{CNP}|\mathsf{NP},\mathsf{APPR})P(\mathsf{CNP} \Rightarrow^* \textit{Arbeit und Geld}) \\ & \cdot P(\mathsf{VAFIN}|\mathsf{APPR},\mathsf{CNP})P(\mathsf{VAFIN} \rightarrow \textit{wird}) \\ & \cdot P(\mathsf{PP}|\mathsf{CNP},\mathsf{VAFIN})P(\mathsf{PP} \Rightarrow^* \textit{von den 37 beteiligten Vereinen}) \\ & \cdot P(\mathsf{VVPP}|\mathsf{VAFIN},\mathsf{PP})P(\mathsf{VVPP} \rightarrow \textit{aufgebracht}) \\ & \cdot P(\$|\mathsf{PP},\mathsf{VVPP}). \end{aligned}$$

Start and end of the path are indicated by a dollar sign ($). This path is very close to the correct structure for layer 1. The CNP and PP are correctly recognized. Additionally, the best path correctly predicts that APPR, VAFIN and VVPP should not be attached in layer 1. The only error is the NP *ein enormer Posten.* Although this on its own is a perfect NP, it is not complete because the PP *an Arbeit und Geld* is missing. ART, ADJA and NN should be left unattached in this layer in order to be able to create the correct structure at higher layers.

4.6.2 The Method

The standard Viterbi algorithm needs to be modified in order to process Markov Models operating on the presented type of lattices. In part-of-speech tagging, each hypothesis (a tag) spans exactly one word. Now, a hypothesis can span an arbitrary number of words, and a span can represent an arbitrary number of alternative word or phrase hypotheses. A state of a Markov Model is allowed to emit a context-free partial parse tree, starting with the represented non-terminal symbol, yielding part of the sequence of words. This is in contrast to standard Markov Models. There, states emit atomic symbols. Note that an edge in the lattice is represented by a state in the corresponding Markov Model. Figure 4.7 shows the part of the Markov Model that represents the best path in the lattice of figure 4.8.

The equations of the Viterbi algorithm (see page 19) are adapted to process a language model operating on a lattice. Instead of the words, the gaps between the words are numbered (see figure 4.8), and an edge between two states can span one or more words, such that an edge is represented by a triple $\langle t, t', q\rangle$, starting at t, ending at t' and representing state q.

We use accumulators $\Delta_{t,t'}(q)$ that collect the maximum probability of state q covering words from position t to t'. These utilize the accumulators δ for context-free grammars as presented on page 23.
Initialization:

$$\Delta_{0,t}(q) = P(q|q_s)\delta_{0,t}(q) \tag{4.21}$$

Recursion:

$$\Delta_{t,t'}(q) = \max_{\langle t'',t,q'\rangle \in \text{Lattice}} \Delta_{t'',t}(q')P(q|q')\delta_{t,t'}(q), \quad \text{for } 1 \le t < T, \tag{4.22}$$

Termination:

$$\max_{Q \in \mathcal{Q}^*} P(Q, \text{Lattice}) = \max_{\langle t,T,q\rangle \in \text{Lattice}} \Delta_{t,T}(q)P(q_e|q). \tag{4.23}$$

Additionally, one has to keep track of the elements in the lattice that maximized each $\Delta_{t,t'}(q)$. When reaching time T, we get the best last element in the lattice

$$\langle t_1^m, T, q_1^m\rangle = \operatorname*{argmax}_{\langle t,T,q\rangle \in \text{Lattice}} \Delta_{t,T}(q)P(q_e|q). \tag{4.24}$$

Setting $t_0^m = T$, we collect the arguments $\langle t'', t, q'\rangle \in$ Lattice that maximized equation 4.22 by walking backwards in time:

$$\langle t_{i+1}^m, t_i^m, q_{i+1}^m\rangle = \operatorname*{argmax}_{\langle t'',t_i^m,q'\rangle \in \text{Lattice}} \Delta_{t'',t_i^m}(q')P(q_i^m|q')\delta_{t_i^m,t_{i-1}^m}(q_i) \quad \text{for } i \ge 1, \tag{4.25}$$

until we reach $t_k^m = 0$. Now, $q_1^m \dots q_k^m$ is the best sequence of phrase hypotheses (read backwards). This again is an instance of dynamic programming.

In the following, we determine the time complexity of the algorithm for three different cases.

A complete lattice is given. If we assume that a lattice for a layer together with the corresponding δ's are given (like the one in figure 4.8), then the computation time needed for this adapted version of the Viterbi algorithm calculating Δ is unchanged compared to the original version. It is linear in the number of words T since we calculate a set of Δ's at each position t, and it is quadratic in the maximum number A of parallel hypotheses starting or ending at the same position (ambiguity rate) since the recursion formula takes into account the combination of all edges ending and starting at some point t. Therefore, the time complexity is $O(A^2T)$.

A unique lower layer is given. This is the situation that occurs during corpus annotation. Part of the structure is fixed and we are looking for a new element in the annotation. First, all matching rules are added, which in the worst case is equal to the number of rules R in the grammar at each of the T positions. Now, a complete lattice is specified and processing is euqivalent to the previous case, resulting in a time complexity of $O(A^2T)$, having $A \leq R$. For practical cases, A is much smaller than R.

All hypotheses of all lower layers are passed to the next layer. This fills the lattice like a chart in standard chart parsing. Each type of phrase can start at any position and end at any position, yielding $O(PT^2)$ elements in the lattice, P is the number of phrase types. There are at most $A \leq PT$ parallel hypotheses starting or ending at the same position, yielding a time complexity of $O(P^2T^2 \cdot T) = O(P^2T^3)$, thus it is cubic in the length of the sequence.

When restricting the parse to non-recursive structures (in the application of chunking), the worst-case complexity is again linear in the length of the sequence. Except for short sequences of words, these non-recursive structures cannot span the entire sequence. The maximum length S of a span that can be covered by a non-recursive structure is determined by the grammar. At each position at most $A \leq PS$ different hypotheses can start or end. Therefore, the time complexity for

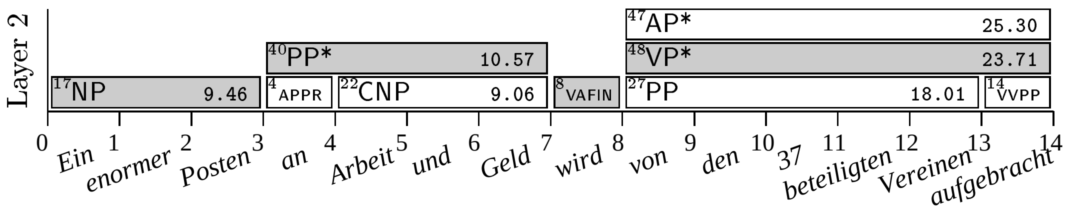

Layer 2
17NP 9.46
4APPR
40PP* 10.57
22CNP 9.06
8VAFIN
47AP* 25.30
48VP* 23.71
27PP 18.01
14VVPP
0 Ein 1 enormer 2 Posten 3 an 4 Arbeit 5 und 6 Geld 7 wird 8 von 9 den 10 37 11 beteiligten 12 Vereinen 13 aufgebracht 14

Figure 4.9: Phrase hypotheses according to a context-free grammar for the second layer, given the results of the first layer in figure 4.8. Phrases that are newly introduced at layer 2 are marked with an asterisk (*). The others were passed up from lower layers. The best path is marked grey.

the chunking task is $O(P^2S^2T)$. Although S can be very large, we usually find that only very short spans can be covered by a chunk in practical applications, so A is much smaller than PS.

We use the Markov Models as filters. At each layer, a subset of hypotheses is chosen according to the assigned probabilities and passed to the next layer. This may change the language that is recognized by the underlying context-free grammar. The influence can be visualized if all discarded hypotheses are marked but kept in the lattice. Parsing proceeds as in chart parsing. In the end, only parses that do not contain marked elements are chosen. It may happen that all complete parses contain at least one marked element. Therefore, the recognized language is either equal to or a subset of the language recognized by the context-free grammar. Furthermore, the number of parses for ambiguous sequences may be reduced for the same reasons.

We have presented accumulators for a bigram model, i.e., one state in the model corresponds to one edge in the lattice. For a trigram model, pairs of edges in the lattice need to be combined to form one state in the Markov Model. The equations become slightly more complex because the Δ's are computed for pairs of edges, and computation time is cubic in the maximum number of parallel hypotheses A.

The process can move on to layer 2 after the first layer is computed. The results of the first layer are taken as the base and all context-free rules that apply to the base are retrieved. These again form a lattice and we can calculate the best path for layer 2.

The Markov Model for layer 1 operates on the output of the Markov Model for part-of-speech tagging, the model for layer 2 operates on the output of layer 1, and so on. Hence the name of the processing model: Cascaded Markov Models.

It often occurs that a phrase hypothesis should be ignored because the hypothesis covers only part of the correct phrase. Looking at figure 4.8, we see that the best hypothesis for range 0 to 3 (*Ein enormer Posten*) is NP #16. Although these three words form an NP, they should be left unattached at this layer because the PP *an Arbeit und Geld* is also part of the NP and the introduction of the NP has to wait until the PP is built. This is handled by keeping more than the one hypothesis that is ranked highest (see chapter 5).

Similarly, it can occur that the path containing the unattached constituents is ranked higher than the combined phrase according to the probability model. For the example above, another parameter setting (learned from another corpus) could assign a higher probability to the sequence ${}_0\mathsf{ART}_1\mathsf{ADJA}_2\mathsf{NN}_3$ than to ${}_0\mathsf{NP}_3$. While this is the desired result for this example because the following PP should also be part of the NP, it may result in unnecessarily incomplete structures. This problem is also addressed in chapter 5. The solution is unification of n-best structures.

4.6.3 Selecting the Best Phrase Hypothesis

We now consider the problem of selecting the best new phrase hypothesis $\langle t, t', q\rangle$, starting at position t, ending at t', representing state q, in the lattice for layer d. The best hypothesis is subsequently added to an existing structure. Remember that a lower layer passes its best phrase hypotheses to the next higher layer. The higher layer adds new phrase hypotheses according to stochastic context-free grammar rules. It is one (or several) of these new phrases that we want to select.

The probability of a phrase hypothesis can in principle be calculated by summing over all paths in the lattice that contain $\langle t, t', q\rangle$:

$$P(\langle t, t', q\rangle) = \sum_{path \in lattice, \langle t, t', q\rangle \in path} P(path) \tag{4.26}$$

The formula is not suitable for practical purposes because of the large number of paths which grows exponentially with the length of the parsed sequence. Again, dynamic programming solves this problem. For each element $\langle t, t', q \rangle$ in a given lattice, the sum is calculated in time linear to the length of the sequence. A version of the Forward-Backward Algorithm is adapted to lattices of phrase hypotheses.

Forward probabilities for a lattice are as follows. The accumulators $\alpha_{t,t'}(q)$ collect the probabilities of all paths from position 0 up to position t and then traversing edge $\langle t, t', q \rangle$. Again, we use $\delta_{t,t'}(q)$ from equation 2.13 on page 23. $P(\mathbf{O})$ is the probability of observing the terminal sequence that gave rise to the lattice. This is the sum of all paths through the lattice.
Initialization:

$$\alpha_{0,t}(q) = P(q|q_s)\delta_{0,t}(q) \tag{4.27}$$

Recursion:

$$\alpha_{t,t'}(q) = \sum_{\langle t'',t,q' \rangle \in \text{Lattice}} \alpha_{t'',t}(q')P(q|q')\delta_{t,t'}(q), \quad \text{for } 1 \leq t < T, \tag{4.28}$$

Termination:

$$P(\mathbf{O}) = \sum_{\langle t,T,q \rangle \in \text{Lattice}} \alpha_{t,T}(q)P(q_e|q). \tag{4.29}$$

The backward probabilities β are defined similarly, but starting at the end of the sequence and running backwards. The accumulators $\beta_{t',t}(q)$ collect the probabilities of all paths from position T backwards to position t and then traversing an edge that represents q. Note that the second index of β, t', is actually not used in the formula, but we add it here to make β parallel to α and to indicate that the accumulators are associated with edges in the lattice.
Initialization:

$$\beta_{t,T}(q) = P(q_e|q) \tag{4.30}$$

Recursion:

$$\beta_{t',t}(q) = \sum_{\langle t,t'',q' \rangle \in \text{Lattice}} \beta_{t,t''}(q')\delta_{t,t''}(q')P(q'|q), \quad \text{for } T > t \geq 1, \tag{4.31}$$

Termination:

$$P(\mathbf{O}) = \sum_{\langle 0,t,q \rangle \in \text{Lattice}} \beta_{0,t}(q)\delta_{0,t}(q)P(q|q_s). \tag{4.32}$$

Note that

$$P(\mathbf{O}) = \sum_{t,q:\langle t,t',q\rangle \in \text{Lattice}} \alpha_{t,t'}(q)\beta_{t,t'}(q) \tag{4.33}$$

for fixed t'.

The set of forward-backward variables is defined as:

$$\gamma_{t,t'}(q) = \frac{\alpha_{t,t'}(q)\beta_{t,t'}(q)}{P(\mathbf{O})} \tag{4.34}$$

This is the probability of traversing edge $\langle t', t, q\rangle$ in the lattice conditional on the observed sequence. Thus, the edge in the lattice with the highest probability is calculated by

$$\operatorname*{argmax}_{\langle t,t',q\rangle \in \text{Lattice}} \gamma_{t,t'}(q) = \operatorname*{argmax}_{\langle t',t,q\rangle \in \text{Lattice}} \frac{\alpha_{t,t'}(q)\beta_{t,t'}(q)}{P(\mathbf{O})} \tag{4.35}$$

Since $P(\mathbf{O})$ is constant for a given lattice, calculations are simplified by

$$\operatorname*{argmax}_{\langle t,t',q\rangle \in \text{Lattice}} \gamma_{t,t'}(q) = \operatorname*{argmax}_{\langle t',t,q\rangle \in \text{Lattice}} \alpha_{t,t'}(q)\beta_{t,t'}(q). \tag{4.36}$$

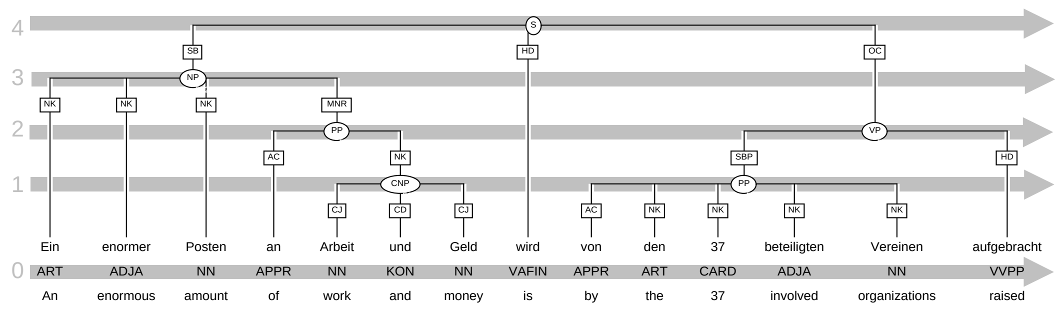
The time complexity of calculating $\alpha_{t,t'}(q)$, $\beta_{t,t'}(q)$, and $\gamma_{t,t'}(q)$ for a lattice is equivalent to the complexity of calculating Δ (see page 68).

4.6.4 Parameter Generation

Parameters for each layer of Cascaded Markov Models are generated separately. Training on annotated data is straight forward. First, we number the layers, starting with 0 for the part-of-speech layer. Subsequently, information for the different layers is collected.

Each sentence in the corpus represents one training sequence for each layer. This sequence consists of the tags or phrases at that layer. If a span is not covered by a phrase at a particular layer, we take the elements of the highest layer below the actual layer. Figure 4.11 shows the training sequences for layers 0 – 4 generated from the sentence in figure 4.10. Each sentence gives rise to one training sequence for each layer.

The context-free rules that are associated with non-terminals nodes represent the outputs of the generated Markov Model, the states are associated with non-terminal


4
3
2
1
0
S
SB
HD
OC
NP
NK
NK
NK
MNR
PP
AC
NK
VP
SBP
HD
CNP
PP
CJ
CD
CJ
AC
NK
NK
NK
NK
Ein enormer Posten an Arbeit und Geld wird von den 37 beteiligten Vereinen aufgebracht
ART ADJA NN APPR NN KON NN VAFIN APPR ART CARD ADJA NN VVPP
An enormous amount of work and money is by the 37 involved organizations raised

'A large amount of money and work is raised by the involved organizations'

Figure 4.10: Example sentence and annotation. The structure consists of terminal nodes (words and their parts-of-speech), non-terminal nodes (phrases) and edges (labeled with grammatical functions). Layers 0 to 4 are indicated by arrows from left to right.

Layer	Sequence
4	S
3	NP VAFIN VP
2	ART ADJA NN PP VAFIN VP
1	ART ADJA NN APPR CNP VAFIN PP VVPP
0	ART ADJA NN APPR NN KON NN VAFIN APPR ART CARD ADJA NN VVPP

Figure 4.11: Training sequences for layers 0 – 4 generated by the sentence in figure 4.10. These plus the corresponding outputs consisting of context-free rules are used to train the Markov Models.

symbols, or pairs of symbols in case of a trigram model. Parameter estimation is done in analogy to models for part-of-speech tagging, and the same smoothing techniques can be applied.

Chapter 5

Applications of Cascaded Markov Models

Chapter Summary

This chapter presents two applications of Cascaded Markov Models. The first one is interactive corpus annotation, i.e., a human annotator and an automatic process incrementally and alternatingly create syntactic structures. The second application is partial parsing. We aim at recognizing structures of fixed depth, where each layer of the structure is represented by its own Markov Model.

5.1 Interactive Corpus Annotation

The techniques of part-of-speech tagging, tagging grammatical functions and phrase categories, and Cascaded Markov Models are employed in semi-automatic corpus annotation. They are integrated into a graphical structural editor (Plaehn, 1998). Automatic and manual processing are interleaved. The graphical editor supports structural manipulations like grouping, ungrouping, attachment, re-attachment, etc., and interacts with a parser that runs in the background. Changes that are made by the annotator are sent to the parser that returns a new phrase hypothesis, labels for the grammatical functions, and their probabilities.

5.1.1 Interleaved Automatic and Manual Annotation

Figure 5.1 illustrates the alternating automatic and manual process. First, part-of-speech tags are inserted by a part-of-speech tagger (see section 4.2). Unreliable

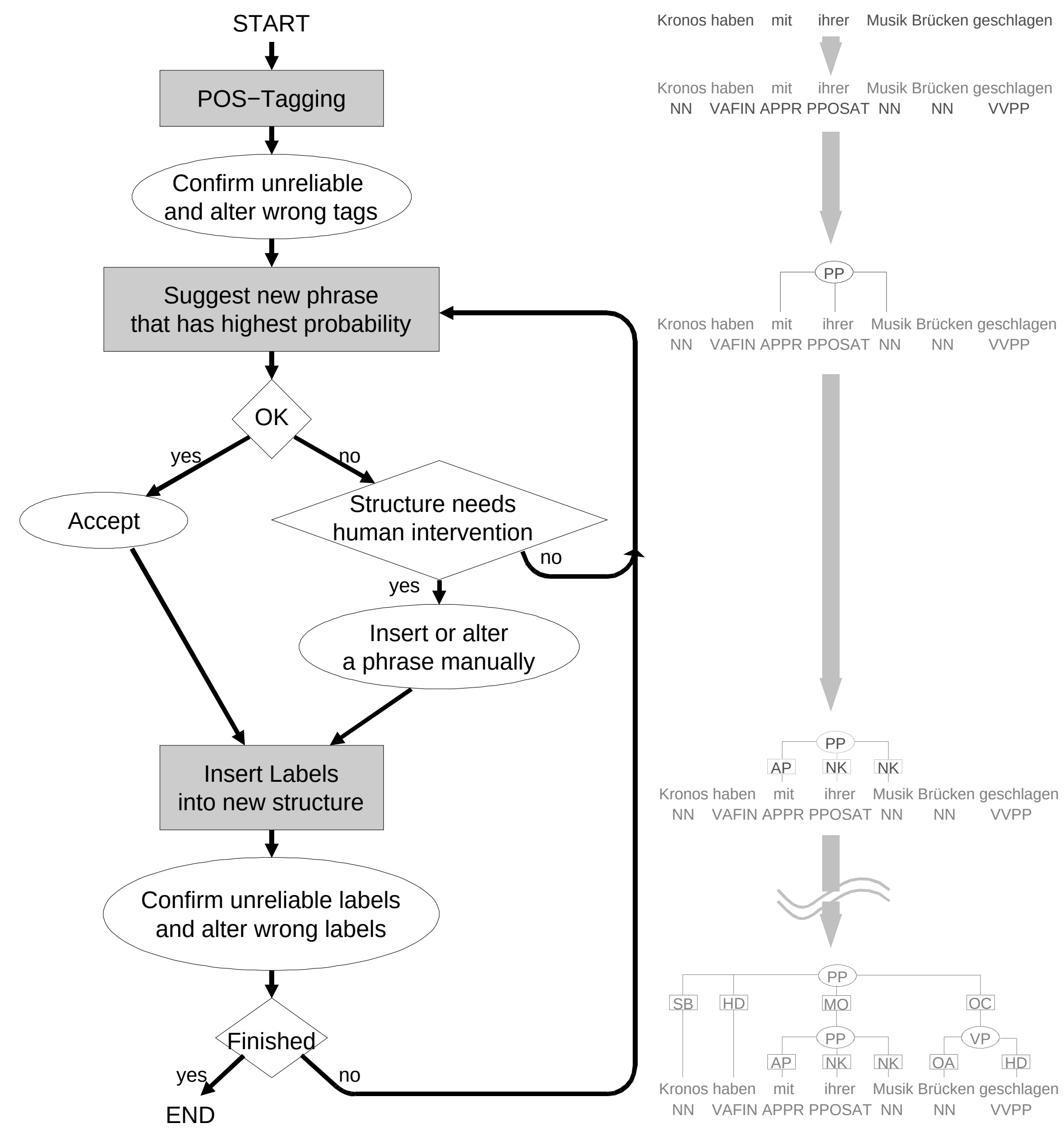

START
POS−Tagging
Confirm unreliable and alter wrong tags
Suggest new phrase that has highest probability
OK
yes
no
Accept
Structure needs human intervention
no
yes
Insert or alter a phrase manually
Insert Labels into new structure
Confirm unreliable labels and alter wrong labels
Finished
yes
no
END
Kronos haben mit ihrer Musik Brücken geschlagen
NN VAFIN APPR PPOSAT NN NN VVPP
PP
AP NK NK
SB HD MO OC
VP
OA HD

Figure 5.1: Interaction of automatic processing (grey boxes) and manual intervention/manual annotation (white ellipses) based on the annotator's decisions (white rhombs).

assignments need to be confirmed by the annotator (see section 4.3). Additionally, the annotator can correct wrong tag assignments. Now, the process of annotation enters a loop that incrementally builds the structure bottom-up. The program suggests a new phrase (i.e., a new structural element and a phrase category) that has the highest probability according to the Cascaded Markov Models (see section 4.6). If the annotator accepts the structure, the program calculates the most probable grammatical functions for the children of the new phrase (see section 4.4). The technique of tagging phrase categories is used to estimate the reliability level of the phrase label (see section 4.5). The process stops if the sentence is completely annotated. Otherwise it proceeds by suggesting a new phrase.

The annotator can reject a suggested phrase. Rejection removes the suggested phrase from the lattices that are used by Cascaded Markov Models. This means that the rejected phrase will not be suggested again for this sentence. Probabilities of all other elements in the lattices are re-calculated. The annotator has two options. He either simply proceeds with semi-automatic annotation and lets the program make another suggestion (the one that has the highest probability after re-calculation), or he decides that the sentence is too difficult for the program and manually inserts or alters a structural element. This triggers automatic insertion of labels for the created or altered phrase, and processing continues with confirming unreliable labels or changing wrong labels.

A typical order of creating a structure is indicated in figure 5.2. Working from left to right and from the bottom to the top, the first node created is the coordinated noun phrase (CNP) *Arbeit und Geld*. The next phrase that is created is the PP *an Arbeit und Geld*, etc. This is a somewhat idealized view since an annotator possibly annotates part of a structure first, and later decides to add one or more elements, or to re-attach some elements. Although we do not want to create the structure in exactly the same order as a human annotator does, we want to simulate the bottom-up construction layer by layer, such that a human annotator has the possibility to supervise the annotation and to correct the structure as soon as possible if the automatic process introduces an error.

In order to verify an automatically created structure, a human annotator needs to

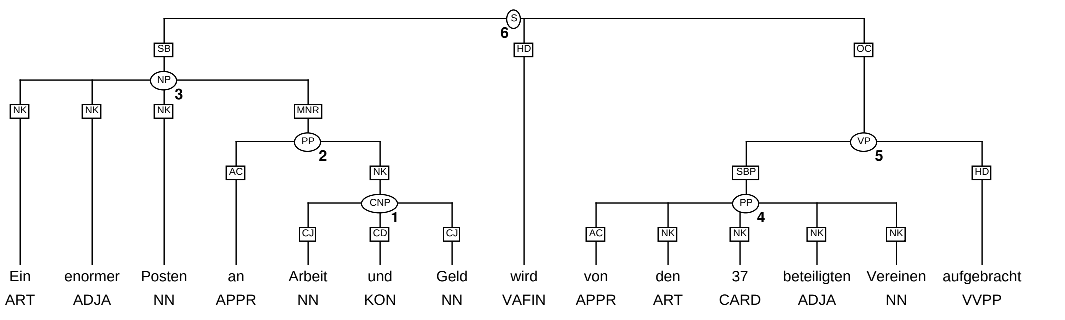


Figure 5.2: Example sentence. The indices at the non-terminal nodes indicate the order in which the nodes are generated during manual annotation.

systematically check all structural elements and all labelings. This is a very difficult and at the same time tedious task because it is easy to miss a small but wrong part. The process is sped up and at the same time made more reliable by presenting the structure incrementally, waiting for confirmation of each increment, and requiring additional actions in case of assignments that are classified as unreliable by the parser.

If the automatic process suggests a wrong element, the annotator rejects that increment, and the program re-calculates probabilities and proceeds with the next-best hypothesis. If the correct element cannot be found by the program, it is manually inserted. The manual element is added to the lattice, all non-compatible elements are removed, probabilities are re-calculated, and the next best hypothesis is suggested.

This mode of annotation is very fast. The graphical user interface combined with Cascaded Markov Models that run in the background are used in the NEGRA project to annotate German newspaper texts, and in the Verbmobil project to annotate German and English transliterated dialogues. We measured the annotation speed of sentences in the NEGRA project. Trained annotators need on average 50 seconds to annotate a sentence with an average length of 17.5 tokens, which is equivalent to approx. 1,300 tokens per hour. This is faster than the annotation speed reported for the Penn Treebank (800 – 1,000 tokens per hour). Additionally, the annotation in the NEGRA project is more detailed, which makes annotation more difficult. Each edge of a NEGRA structure is labeled with one of 45 grammatical functions.

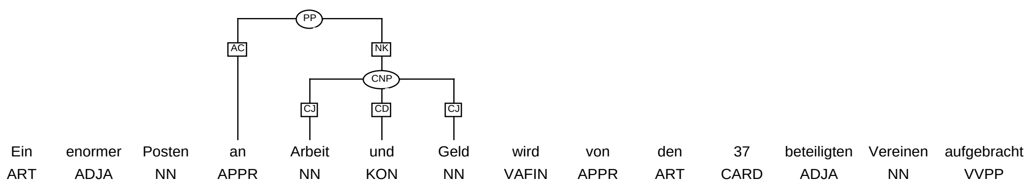


Figure 5.3: Structure after annotating two phrases, one at level 1 (CNP) and one at level 2 (PP).

5.1.2 Selecting the Best Phrase Hypothesis

Cascaded Markov Models are used in corpus annotation to select the *best new phrase given a partial analysis.* A structural analysis is created phrase by phrase such that a human annotator can follow the parsing process and can intervene if necessary. At each point, the best new phrase, i.e., the phrase with the highest probability according to the model, is added to the existing structure and presented to the annotator. The annotator's task is to accept or reject the new phrase. The stepwise presentation of phrases guides the annotator through the structure and facilitates the detection of errors.

At each point, we use the structure annotated so far to generate a new phrase. The given structure typically consists of parts that are automatically created, and other parts that are manually added or altered. This structure is divided into levels, and for each level the corresponding lattice is created.

We use the structure that is given in figure 5.3 as an example. Two phrases are already annotated: the coordinated noun phrase (CNP) at level 1 and the PP at level 2. These two elements and the part-of-speech tags that are generated at layer 0 are filled into the lattices for layer 1 and 2 as shown in figure 5.4. These form *trivial lattices*, consisting of just one path. We need an additional layer 3 in order to create new structural elements on top of the existing ones. This new layer is initialized with the elements of the layer below.

After having built the trivial lattices that consist only of the given structural elements, we add phrase hypotheses according to context-free rules that are learned from a corpus. There is a rule NP $\rightarrow$ ART ADJA NN, thus the corresponding

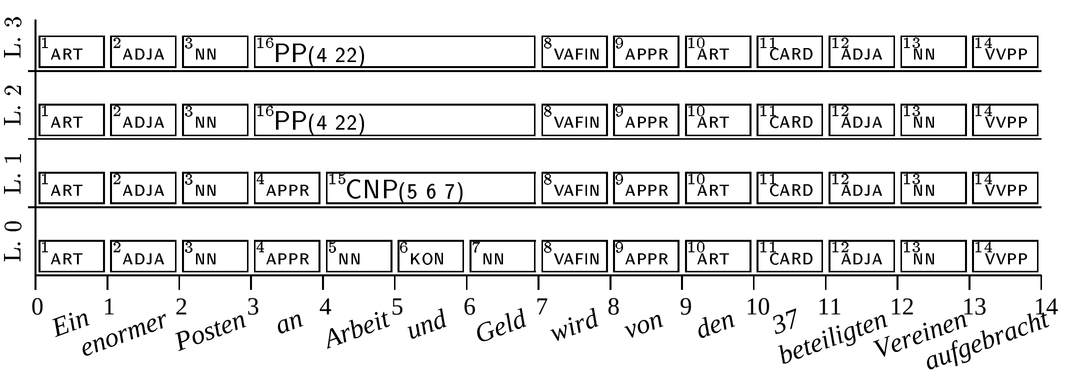


Figure 5.4: Trivial lattices for layers 0 to 3 for the existing structure as given in figure 5.2. Numbers in the upper left corner of each element are identifiers, numbers in brackets to the right of a phrase category point to the children in the lower layer.

hypothesis is added between positions 0 and 3 at level 1. It is added at level 1, because the resulting structure is of depth 1. The grammar also has a rule NP $\rightarrow$ ART ADJA NN PP. This results in a structure of depth 3 for the given example, so it is added to level 3. Figure 5.5 shows the lattices with all phrase hypotheses that can be added at this point. The newly added elements are marked with an asterisk.

No new edges are added to layer 0, since this layer represents all structures of depth 0, i.e., part-of-speech tags. These are fixed during the first iteration cycle for each sentence (nevertheless, the annotator can change them manually at any time).

Several hypotheses are added to layer 1. The best path in the resulting lattice for this layer is marked grey. There are no new edges at layer 2 in the example, because there is no context-free rule that takes the existing structure of depth 1 (the CNP). There are several context-free rules that take the existing structure of depth 2 (the PP) and build a new phrase on top of it; thus, there are new edges at layer 3. The best paths at layers 0 and 2 are trivial since there is just one path. The best path at layer 1 uses two of the new phrase hypotheses: ^{18}NP and ^{24}PP. The best path at layer 3 uses one new element: ^{30}NP.

It may happen that an annotated structure has more levels than the number of Markov Models that we are using in the cascades. In this case, all additional levels are handled by copies of the topmost Markov Model.

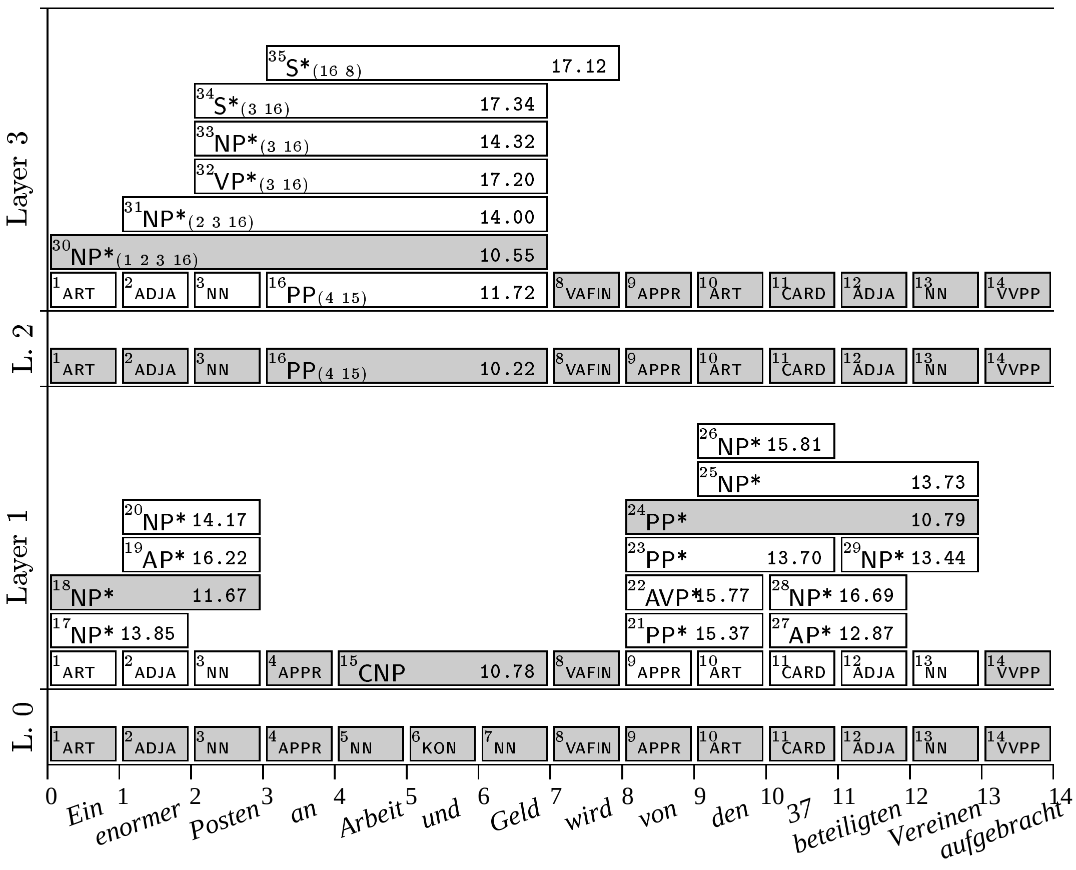


Figure 5.5: New hypotheses are added to the trivial lattices in figure 5.4 according to context-free grammar rules. These are marked with asterisks. Each lattice is processed by a Markov Model. The best path of each lattice is marked grey, the values at the right end of each edge indicate the negative logarithm of the γ values without normalization by $P(\mathbf{O})$. The edge ^{30}NP has the best γ value of all newly added edges. It is therefore disclosed to the annotator (cf. figure 5.6).

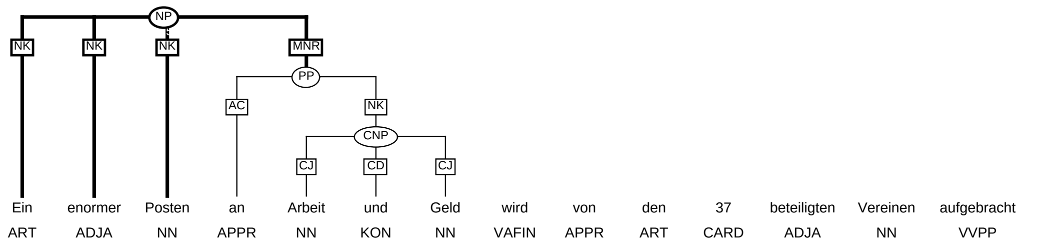


Figure 5.6: Given the annotation of the coordinated noun phrase CNP and the PP, the Cascaded Markov Models suggest that the next phrase is the NP *Ein enormer Posten an Arbeit und Geld.*

We now need to decide which of the added hypothetical phrases should be added to the existing structure that is given in figure 5.3. This is done according to the γ-probabilities presented in section 4.6.3. The phrase with the highest probability is selected[1]. The model that is trained on the NEGRA corpus assigns the highest value to the NP *Ein enormer Posten an Arbeit und Geld* at layer 3, therefore the structure shown in figure 5.6 is presented to the annotator, the newly added phrase is highlighted.

If the annotator decides that this is not a correct phrase, this phrase hypothesis is removed from the lattice, new probabilities are calculated, and the best phrase according to the new probabilities is chosen.

If the annotator decides that this phrase is correct, then the process proceeds by selecting labels for the phrase category (node label) and grammatical functions (edge labels). The presentation of the phrase hypothesis already contains labels. They are the best guess of the program and added at that point for the convenience of the annotator. But they may need closer examination.

5.1.3 Label Selection

After specification of a structural element, two different types of labels are added: the phrase category of the mother node, and the sequence of grammatical functions of the edges.

[1]Highest probabilities correspond to lowest values in figure 5.5 because negative logarithms are shown.

The phrase category is assigned by the tagging technique described in section 4.5. The best label as well as its probability and the probabilities of alternative assignments are calculated. If probabilities are close together, the assignment is regarded as unreliable (see section 4.3) and the annotator is asked for confirmation. If the best label's probability is much larger than the others, that label is assigned without further action of the annotator.

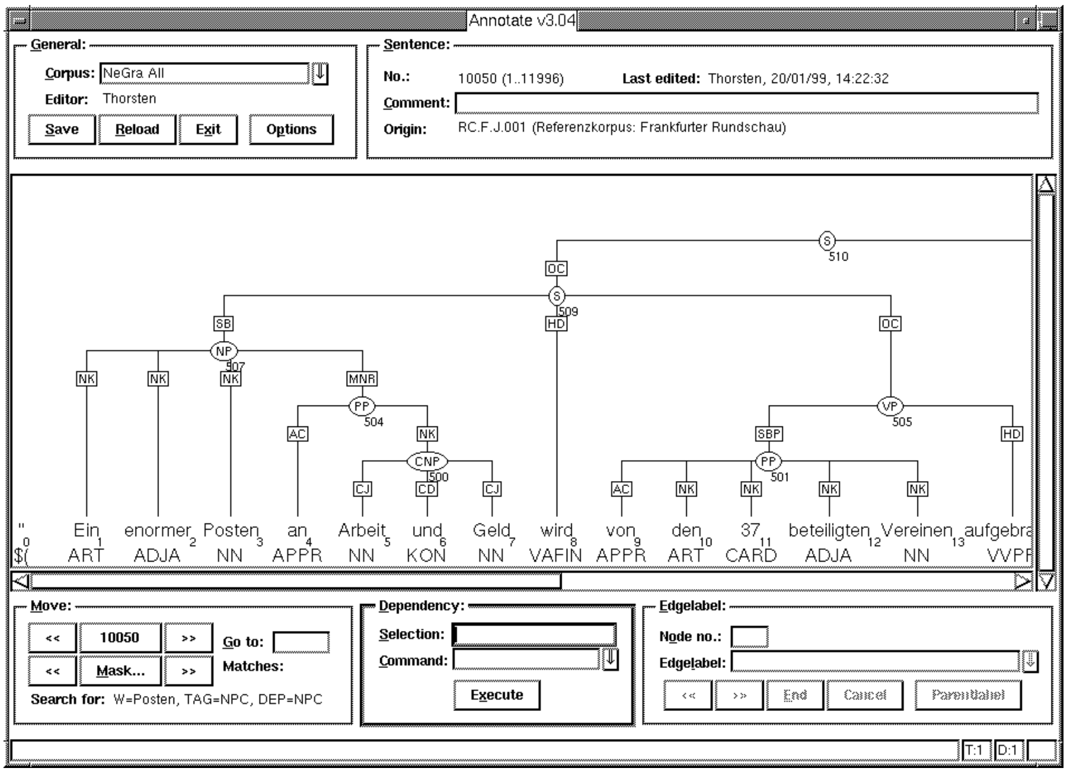
After having determined the phrase category, all grammatical functions within that phrase are assigned according to the tagging technique described in section 4.4. Again, probabilities of alternatives are calculated, and the annotator is asked for confirmation in those cases in which the assignment is classified as unreliable.

After specification of all labels of the new phrase, the annotation process proceeds with a new phrasal element until the structure is complete.

5.1.4 Graphical Annotation Tool

The interactive annotation mode using Cascaded Markov Models is combined with the graphical annotation tool *Annotate* (Plaehn, 1998). Parser and annotation tool are separate programs communicating with each other. The tool sends the already existing part of the annotation to the parser, which either suggests part-of-speech tags (when starting to annotate a sentence) or selects the best phrase hypothesis (cf. section 5.1.2) and sends it to the annotation tool. The parser is informed whether the annotator accepted or rejected the suggestion and accordingly updates the data structures used by the Markov Models and recalculates probabilities.

Figure 5.7 shows a screen shot of *Annotate*. It facilitates the handling of different corpora and tagsets, defines several functions for structure and tag manipulation, and interfaces with the parser. The annotation tool also has an interface to a database for storing the corpus. All functions are accessible by menus, but after some time of training fastest annotation is achieved by a combination of mouse and keyboard input.



Corpus: NeGra All
Editor: Thorsten

No.: 10050 (1..11996) Last edited: Thorsten, 20/01/99, 14:22:32
Comment:
Origin: RC.F.J.001 (Referenzkorpus: Frankfurter Rundschau)

"	Ein	enormer	Posten	an	Arbeit	und	Geld	wird	von	den	37	beteiligten	Vereinen	aufgebra
0	1	2	3	4	5	6	7	8	9	10	11	12	13	
$(	ART	ADJA	NN	APPR	NN	KON	NN	VAFIN	APPR	ART	CARD	ADJA	NN	VVPF

10050
Search for: W=Posten, TAG=NPC, DEP=NPC

Figure 5.7: The graphical tool for corpus annotation *Annotate*. It uses an interface to Cascaded Markov Models that generate the next phrase hypothesis presented to the annotator. Additionally, all necessary manual tree manipulations are supported.

5.2 Partial Parsing with Cascaded Markov Models

We now combine the methods that were introduced in chapter 4 to form a partial parsing model that is based on Markov Models. Parsing as presented here is an extension of part-of-speech tagging, or, looking at the model, part-of-speech tagging is a special case of parsing with Cascaded Markov Models.

5.2.1 The Layered Partial Parsing Model

The proposed model utilizes Cascaded Markov Models. Starting with a Markov Model for part-of-speech tagging, which is just a special case of the next layers, it builds up the structure layer by layer, leaving selected ambiguity in the structure while removing all hypotheses that can be excluded with a pre-defined reliability. Probabilities are based on Markov Models that process layers of phrase and part-of-speech categories, and on stochastic context-free grammar rules. The layers are stored in a compact format, a lattice. Some of the hypotheses within each layer are selected based on their probabilities and serve as input for the next higher layer. The number of layers is fixed in advance and each layer is processed by a separate Markov Model.

Figure 5.8 shows an overview of the model. Processing starts with an input sequence of words. It is not required to form a sentence, but can be also an isolated NP or any type of partial input as such fragments regularly occur in corpus data.

The first process, at layer 0, is part-of-speech tagging. The tagger selects the best tag and all close competitors for each word, and passes them to the next layer. To keep the figure simple, only best hypotheses are shown.

The next step in processing is the handling of layer 1. This and the subsequent layers are all based on the same principle. First, all hypotheses according to the underlying context-free grammar are retrieved. Among those is the rule NP $\rightarrow$ APPR PPOSAT NN in the example. The hypotheses form a lattice that is processed by the Markov Model for layer 1. The model operates from left to right which is an important addition to context-free grammars. So we do not only take into account probabilities of nodes generating some set of children, but also transitional

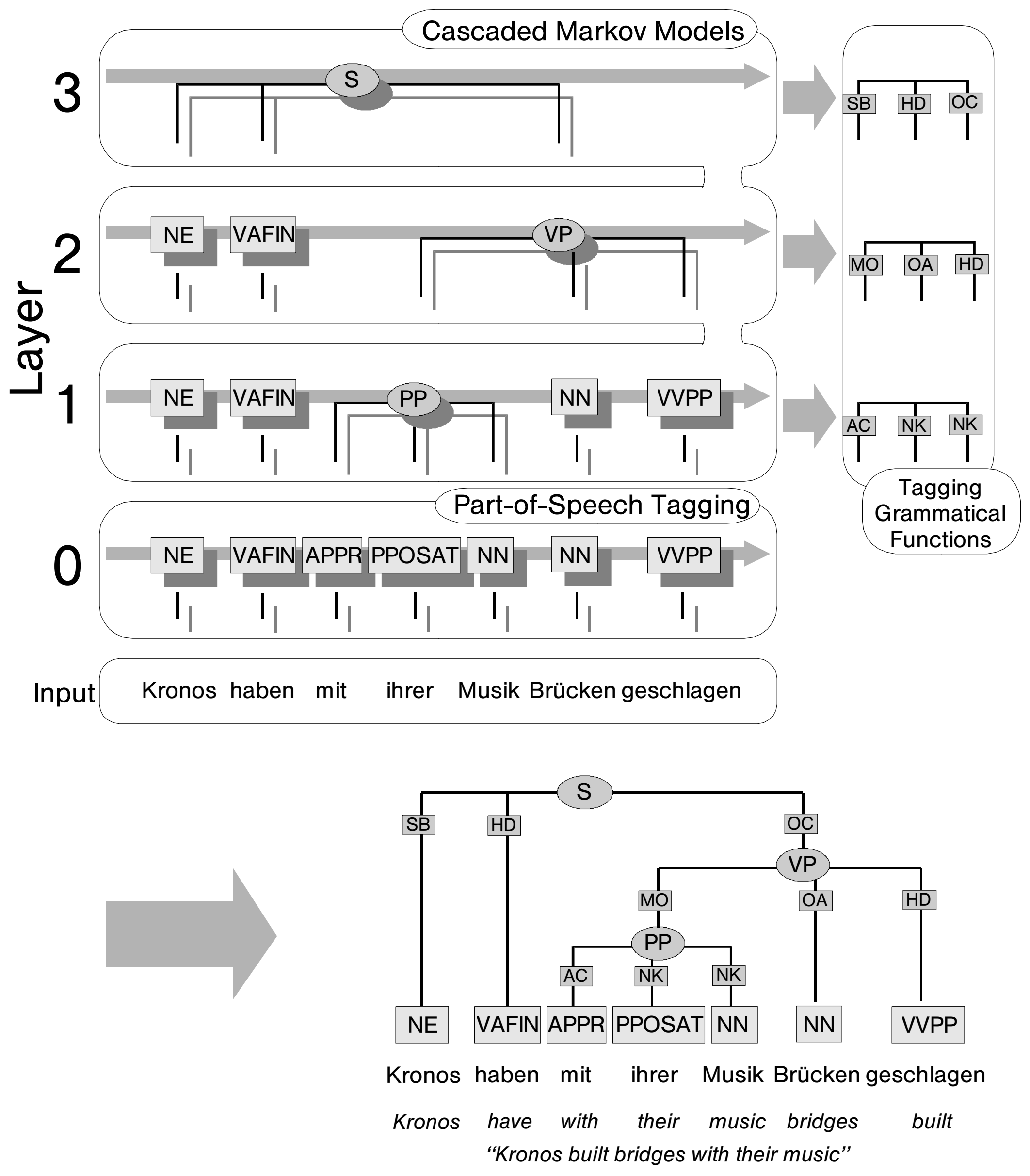

Cascaded Markov Models
Layer
3
2
1
0
S
NE
VAFIN
VP
PP
NN
VVPP
APPR
PPOSAT
SB
HD
OC
MO
OA
AC
NK
Part-of-Speech Tagging
Tagging Grammatical Functions
Input
Kronos haben mit ihrer Musik Brücken geschlagen
Kronos have with their music bridges built
"Kronos built bridges with their music"

Figure 5.8: The combined, layered processing model. Starting with part-of-speech tagging (level 0), possibly ambiguous output together with probabilities is passed to higher levels (only the best hypotheses are shown for clarity). At each level, new phrases and grammatical functions are added.

probabilities between terminal and non-terminal nodes that are not necessarily sister nodes, e.g., the transitions from the finite verb (VAFIN) *haben* to the PP *mit ihrer Musik*, which are not sister nodes in the resulting structure. Recognized phrases are processed by the tagger for grammatical functions before they are passed to the next layer.

By passing more than one hypothesis to the next layer, it can select the hypothesis that fits best into its context even if it is not the best hypothesis at the lower level. This mechanism permits interaction between the levels: the hypotheses are generated and filtered bottom-up, the final decision is made top-down.

Each layer of the resulting structure is represented by its own Markov Model. As a consequence, only structures up to a fixed depth can be recognized. If we run n Markov Models, only structures up to depth n are recognized, all phrases at layer n stay un-attached. The best hypothesis (or the best sequence of hypotheses) at the highest layer determines the recognized structure.

Using a fixed number of layers gives up the power of context-free grammars and restricts the recognized sequences to regular languages. But looking at parsed corpora, the average depth of structures is very small (Penn Treebank: 9.1; Verbmobil English: 4.8; NEGRA corpus: 4.0; Verbmobil German: 3.8)[2]. Even if we want to cover most of the sentences (e.g., 99%), the depth that is needed can easily be represented by the appropriate number of Markov processes (Penn Treebank: 20; Verbmobil English: 10; NEGRA corpus: 9; Verbmobil German: 8). Processing 99% of all sentences correctly in complex domains is still a distant goal for any current method of generating treebank structures. Thus, we do not think that concentrating on regular languages *during processing* is an unfortunate restriction; it is an advantage for the parsing process. The grammar is still expressed by context-free means, but recursion is restricted during processing and context information is added.

An alternative solution is the application of the top-layer Markov Model to all higher layers, such that the depth of parsed structures is not restricted. This yields a complete context-free parsing model that either uses Markov Models for pruning

[2]The differences in the average depth are due to the different languages as well as to the different annotations schemes favouring deep or flat structures.

or for handling an agenda in an agenda-based parser (see section 8.2).

5.2.2 A Processing Example

Figure 5.9 shows all hypotheses that are generated for the different levels when processing the sentence of figure 5.2 on page 78. Layer 0 is used for part-of-speech tagging. Several words are ambiguous w.r.t. the tagset (*enormer, an, von, den, beteiligten*). The best tags and all tags with probabilities within a pre-defined beam are passed to the next higher level. These are marked grey. There are no alternatives within the beam in this example, so level one starts with a unique sequence of tags.

Layer 1 first adds all phrase hypotheses according to a context-free grammar. As an example, hypothesis #26 is created because the grammar contains the rule NP → ART ADJA NN. The γ value of this hypothesis ($10^{-12.24}$; not normalized) is indicated by the number to the right (cf. section 4.6.3). All other hypotheses and their probabilities are analogously inserted. The 17 hypothetical phrases plus 14 entries for part-of-speech tags form a lattice that has 665 paths from node 0 to node 14.

These paths are evaluated and the hypotheses are ranked by a trigram model as described in section 4.6. According to the model, the best path consists of hypotheses ^{26}NP (*ein enormer Posten*), ^{4}APPR (*an*), ^{31}CNP (*Arbeit und Geld*), ^{8}VAFIN (*wird*), ^{36}PP (*von den 37 beteiligten Vereinen*), and ^{14}VVPP (*aufgebracht*). Elements of the best path and elements with probabilities that are close to those in the best path are marked grey[3]. The threshold is set to $\theta = 100$, thus all hypotheses having a probability greater or equal to one hundredth of the best path's probability are taken into account, the others are pruned.

The selected elements are passed to the next layer. Therefore, all elements that are marked grey at layer 1 can also be found at level 2. The process is repeated at layer two. Again, hypotheses are generated according to context-free grammar rules, and again the resulting lattice is processed. The only difference is that the Markov Model for layer 2 is different from that at layer 1. The indices in brackets

[3]The single best path at this layer is not indicated in the figure. The figure shows the best path at the top-most layer and all children that belong to phrases in this path.

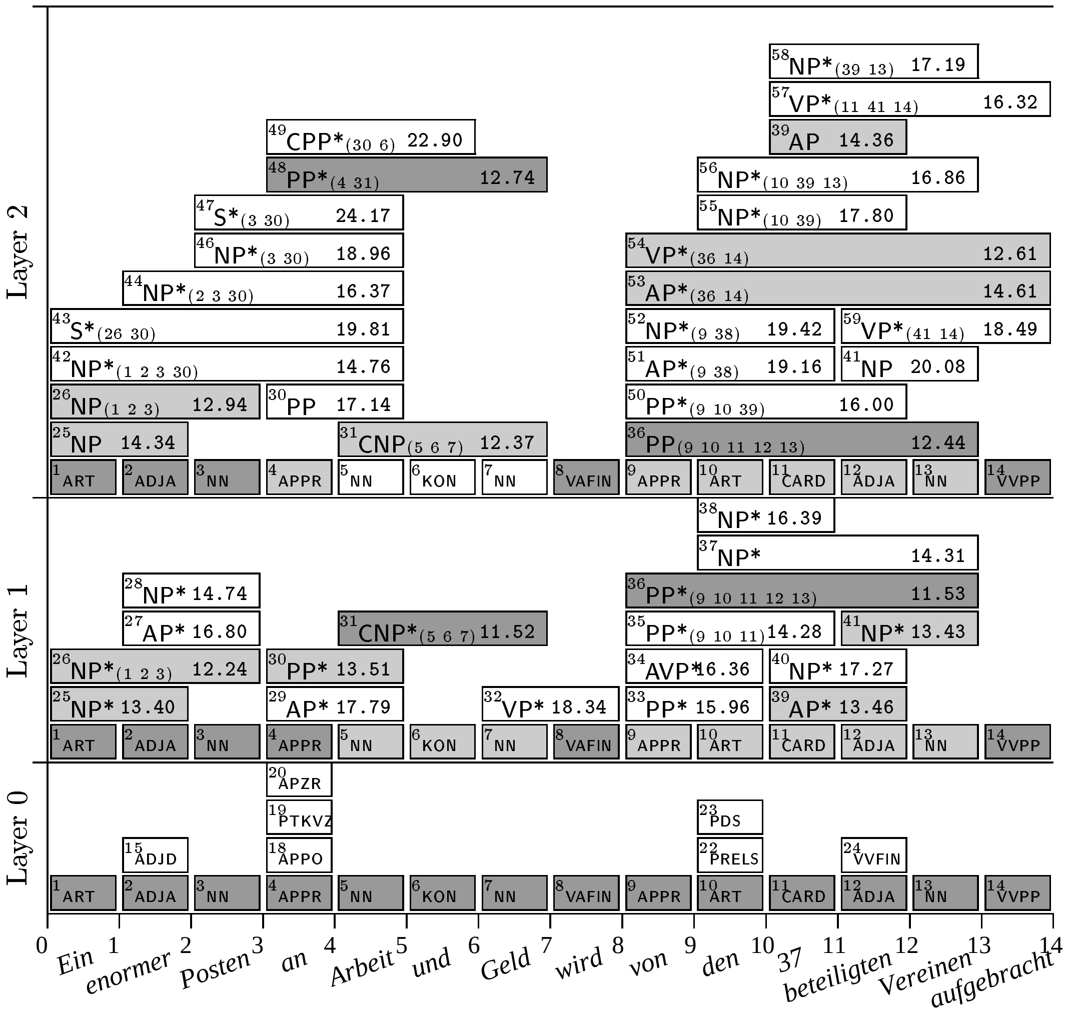


Figure 5.9: Processing of the example sentence. The figure shows all hypothetical phrasal nodes. All grey nodes are passed to the next higher level. Nodes that are newly added at a layer are marked with an asterisk. The dark grey nodes belong to the structure of the best hypothesis at the topmost level (level 4). Each edge has an index in the upper left corner, the indices of children nodes are in brackets (omitted for short nodes), and the number at the right side is the logarithm of the edge's γ value (not normalized).

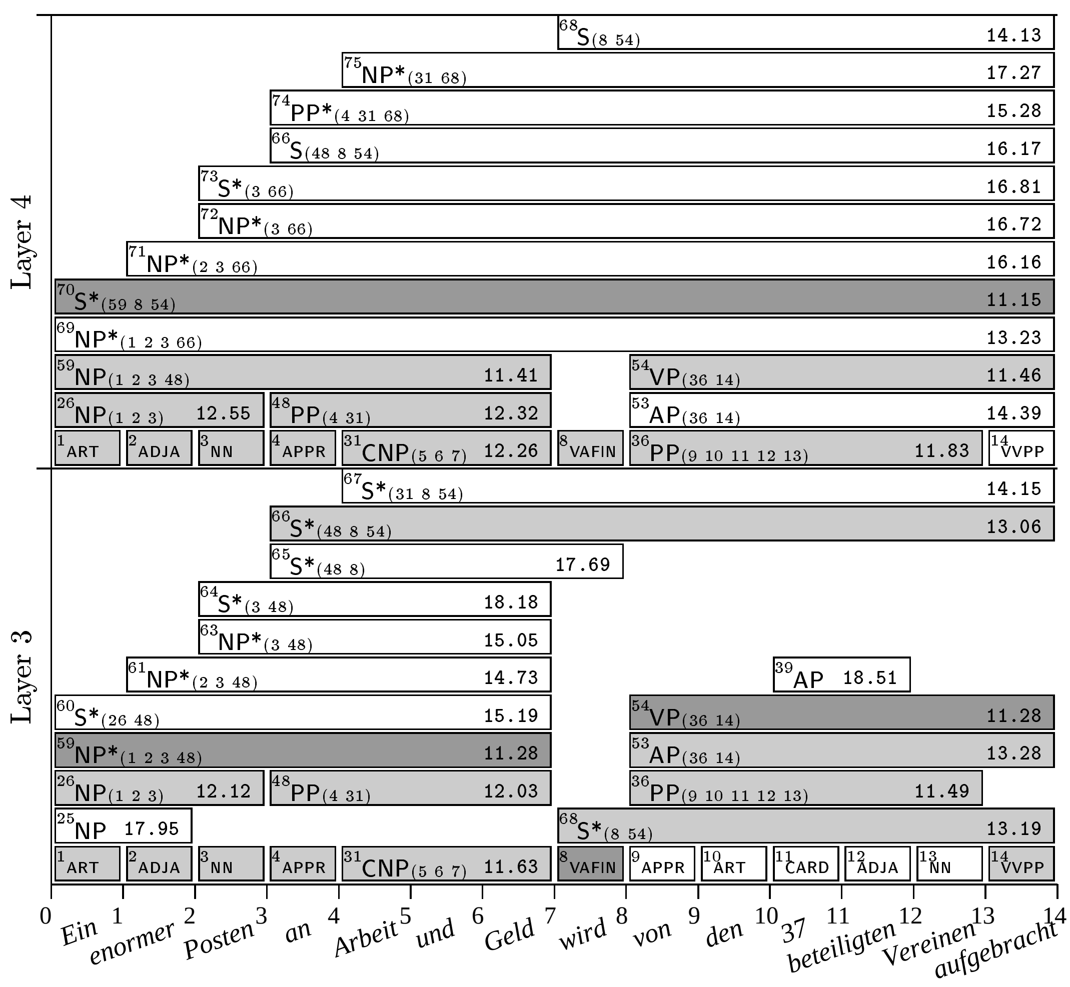


Figure 5.10: Processing of the example sentence (layers 3 and 4)

to the right of the phrase label of elements at layer 2 (and higher) indicate the list of proposed children of the phrase. As an example at level 2, ^{54}VP immediately dominates ^{36}PP and ^{14}VVPP. We need to keep track of the children because there is the possibility that a phrase can be constructed in more than one way. The different sub-structures are distinguished by the list of children.

Cascaded Markov Models utilize context-free rules to generate hypotheses. The final parsing result is identical to that of context-free parsing if we do not run the Markov Model at each level but instead pass all hypotheses to the next higher layer. But this mode would miss important information.

In contrast with a context-free model, Cascaded Markov Models additionally take left-to-right transitional probabilities of nodes into account that are not sister nodes but occur at the same level in the syntactic structure. This is used for a second *horizontal* probability model, in addition to the context-free *vertical* model.

We use the horizontal model as a filter that selects hypotheses based on a local (trigram) context. A large number of hypothetical phrases can be discarded.

The best path of hypotheses is selected at the highest layer (which is layer 4 in the example). The best paths at the lower layers were used to create the lattice at the highest layer. Now that we know the best path at the highest layer, we can collect the corresponding elements and their children which represent the generated structure.

The best path at layer 4 consists of the single hypothesis ^{70}S. This phrase immediately dominates ^{59}NP, ^{8}VAFIN, and ^{54}VP. All elements dominated by the best hypothesis at the topmost layer are marked dark grey in figures 5.9 and 5.10.

The result is a complete parse. Note that Cascaded Markov Models can also generate partial parses. This happens if either the structure for a complete parse has more layers than running Markov Models or if a complete structure is assigned a lower probability than a partial structure.

5.2.3 Finding Top-Level Chunks

It may occur that a parse with greater depth has a lower probability than a partial parse with smaller depth. If, for example, the path consisting of just the edge ^{70}S in figure 5.10, layer 4, had a lower probability than the path ^{59}NP ^{9}VAFIN ^{54}VP, then the latter would be selected as the best path of layer 4. Although this effect is desirable in general, especially at lower levels where the process has the ability to delay an attachment, we do not want it to occur at the topmost layer.

In order to alleviate the problem, we do not simply use the best path of the topmost layer, but n-best paths (or all paths within a pre-defined beam) as long as the structures unify with the structure that is associated with the best path. Two structures unify if a comparison does not yield crossing brackets and if there is a phrase is present in both structures, then it has identical labels and identical sub-structures.

If we assume that ^{59}NP ^{9}VAFIN ^{54}VP is ranked first, and ^{70}S second, then we test if they unify. Since a comparison shows that the brackets do not cross, and that each phrase of the first structure is also present in the second structure with the same labels, they unify, and the resulting structure is that of ^{70}S.

Chapter 6

Evaluation Methodology

Chapter Summary

Evaluation is a central point in the development of language technology. It allows the comparison of different systems and is the basis for improvements. Therefore, this chapter explicitly presents the methodology for evaluating the techniques introduced in this thesis. We determine requirements for an experimental setup and different measures for tagging and parsing.

Evaluating a system that processes natural language usually serves two purposes. The first one is to make predictions about the system's performance when processing new, previously unseen data. The second purpose is to make the system comparable to other systems. An evaluation needs to be carefully constructed in order to match these requirements.

A fair comparison of systems is only possible if they are applied to identical tasks. Small changes in the tagset may have a big influence on the tagging accuracy (Elworthy, 1995). The same is true for the structural encoding. Using different structures may result in significant gain of recall and precision (Johnson, 1998). Different domains also heavily influence the outcome. This means that most reported evaluations are not comparable. Also, different sizes of training corpora influence processing results. At a closer look, subtle differences emerge that may significantly influence the outcome.

Even if two systems process the same language, use the same tagset and the same structural encoding, the numbers given by the authors may reflect different measurements and may therefore not be compared.

Some authors report performance with a "complete lexicon", i.e., all words occuring in the test corpus that are not already in the lexicon are (manually) added before testing. The opposite situation is an "incomplete lexicon", i.e., there may occur unknown words in the test data and the system has an automatic component to handle unknown words. Results obtained with a complete lexicon are not comparable to those obtained with an incomplete lexicon.

Another difference in evaluating parsers is the actual input for the parser. Some systems start with a sequence of words, others additionally need disambiguated parts-of-speech for the words and start parsing on the parts-of-speech.

Several investigations impose restrictions on the tested material. Examples found in the literature are:

- Test data is restricted to sentences completely consisting of words that are among the n most frequent words of the corpus (e.g., $n = 3000, 5000, \ldots$).
- Test data is restricted to sentences of maximum length k (e.g., $k = 20, 30, \ldots$).
- Test data is restricted to sentences without coordination.

Also, the size of the test corpus and the number of iterations are important. From a statistical point of view, a single test run on a few dozen test sentences usually does not yield significant results. A reliable method is to test several times on a large test set and report averaged results as well as standard deviations. This is usally achieved by dividing a corpus into 90% for training and 10% for testing and repeating the experiment 10 times. Each time another 10% of the corpus is used for testing.

6.1 Rules of Evaluation

The golden rule for evaluating natural systems is never to look at test data before actually testing (Magerman, 1994). Do not use it for training, extracting vocabulary, manually creating rules, etc. This is prevalent to make predictions about the system's performance in the "real world", i.e., processing of data that was not seen before. Any violation of this rule inhibits these predictions.

Another rule is that one should not test too many times on the same test set. Improving results on the same test data does not necessarily mean an improvement of the system, but can also be a result of overfitting the test data.

The division of a single corpus into 90% training set and 10% test set, which is a common technique, is put into question by Magerman (1994). He argues that this makes the test data as statistically similar to the training data as possible and thereby inappropriately improves the perceived test performance.

A number of papers (e.g. Sekine, 1997) as well as our own tests on the different domains of the Brown corpus have shown that accuracy decreases if training and test sets are chosen from different domains (compared to the situation of training and testing on the same or very similar domains). The exact difference in accuracy heavily depends on the chosen domains. Different results may also be obtained if training and test material are from the same domain, but from a different *source*. This occurs, e.g., when training on the Wall Street Journal part of the Penn Treebank and testing on some other newspaper.

So, while it is true that a 90%/10% division ensures statistical similarity of training and test part, we argue that this division yields reliable results for a particular *source*. Results do not significantly vary when training on one week of newspaper text and testing on some other week (or month) compared to training and testing on the same week.

The important fact is that the texts come from the same source. This means that it is necessary to indicate the source when reporting on experiments that train and test on partitions of the same corpus, because results cannot be transfered to other sources. The best solution would be to test on a large number of texts from different sources. But this is usually unfeasable because of the unavailability of sufficient amounts of annotated text material.

The main point in evaluation is that a system should be tested in a way that is as close as possible to its intended application, which usually includes newness of processed data and incomplete vocabulary, but also the restriction to a particular domain or even to a particular source.

6.2 Tagging Accuracy

Taggers are usually evaluated in terms of *accuracy* on the basis of words, i.e., the number of correctly assigned tags $f_{correct}$ in relation to the total number of processed words N:

$$ACC = \frac{f_{correct}}{N}$$

If the experiment is repeated k times with different training and test sets, one can calculate the average accuracy $\overline{ACC}$ and the variance s^2:

$$\overline{ACC} = \frac{\sum_{i=1}^{k} ACC_i}{k} \qquad s^2 = \frac{\sum_{i=1}^{k} (ACC_i - \overline{ACC})^2}{k-1}$$

Assuming a normal distribution for the experimental outcomes, these can be used to calculate a confidence interval with confidence degree p. A confidence degree of p means that we have found an interval μ that actually contains the real accuracy value with a chance of p:

$$\mu = \overline{ACC} \quad \pm \quad t\sqrt{\frac{s^2}{k}} \qquad \text{(confidence degree } p\text{)}$$

t is determined from the t-distribution function F with $k-1$ degrees of freedom such that $F(t) = (1+p)/2$. For a confidence degree of 95% and $k = 10$ iterations we find

$$\mu = \overline{ACC} \quad \pm \quad 1.96\sqrt{\frac{s^2}{10}} \qquad \text{(confidence degree 95\%)}$$

If the experiment is run just once, we can estimate the confidence interval by making the assumption that the correct or wrong assignment of tags follows a binominal distribution, such that the assignment is seen as a random sequence of *wrong* and *correct* events. In this case, the 95% confidence interval is

$$\mu = ACC \pm 1.96\sqrt{\frac{ACC(1-ACC)}{N}} \qquad \text{(confidence degree 95\%)}$$

for large values of N. If we find an accuracy of $ACC = 0.96$ for a sample of $N = 10,000$, we estimate

$$\mu = 0.96 \pm 1.96\sqrt{\frac{0.96(1-0.96)}{10,000}} = 96\% \pm 0.2\% \qquad \text{(confidence degree 95\%)}.$$

6.3 Crossing Brackets, Recall and Precision

The *PARSEVAL* scheme (Black, Ezra, et al., 1991) defines three measures to characterize the performance of a parser that produces context-free structures:

Crossing Brackets. This is the average number of constituents (pairs of brackets) per sentence that are proposed by the parser but that violate the constituent structure in the treebank. Given a sentence with k words and gaps numbered from 0 to k, a crossing bracket error occurs if the parser proposes a constituent ranging from i to j, and the treebank contains a constituent ranging from l to m such that

$$i < l < j < m \qquad \text{or} \qquad l < i < m < j.$$

Recall. This is the number n_{corr} of non-terminal nodes proposed by the parser for which there exist a corresponding node in the treebank that covers the same words in relation to the total number of non-terminal nodes in the treebank $n_{treebank}$:

$$\mathit{recall} = \frac{n_{corr}}{n_{treebank}}$$

The measure can be made stricter by requiring that the proposed node should have the same label as the node in the treebank. If there are $n_{labeled\_corr}$ such nodes, we have

$$\mathit{labeled\ recall} = \frac{n_{labeled\_corr}}{n_{treebank}}.$$

Precision. This uses the same quantities n_{corr} and $n_{labeled\_corr}$ as recall but relates them to the total number of nodes n_{parser} proposed by the parser:

$$\mathit{precision} = \frac{n_{corr}}{n_{parser}}$$

$$\mathit{labeled\ precision} = \frac{n_{labeled\_corr}}{n_{parser}}.$$

Calculation of confidence intervals for recall and precision is identical to the case of tagging accuracy.

Partial parsing requires a slight extension of this scheme. Since a partial parser does not aim at the construction of the complete treebank structure but only parts thereof, we need to determine the maximum number of elements which the parser can recognize. If a parser recognizes only NP nodes, the maximum number of recognized nodes is the number of NPs in the treebank. So $n_{treebank}$ is replaced by the *number of NPs in the treebank* to determine recall. The precision formula does not change. Another case which will occur in the evaluation is the restriction of structures to a fixed depth d. In this case, we will use the *number of nodes in the treebank that are within the lowest d levels of the structures* as denominator in the recall formula.

If $n_{partial}$ denotes the number of nodes which the partial parser can recognize, recall for partial parsing is

$$partial\ recall = \frac{n_{corr}}{n_{partial}}$$

$n_{partial}$ depends on the exact task of the partial parser.

6.4 Exact Match

Exact match indicates the percentage of sentences for which structure and tags proposed by the parser, including all tags and labels, are identical to those contained in the treebank. This is a very strict metric, since a single small error, e.g., a wrong label in one of the nodes, spoils the complete structure and has the same effect as a completely nonsensical parse. Therefore, the exact match metric is too coarse grained and usually given in combination with recall and precision.

The *structural match* metric does not require the tags and labels to be identical; thus it measures the percentage of sentences for which the tree structures generated by the parser are identical to the structures in the treebank.

6.5 n-best and Alternative Assignments

Taggers and parsers may assign more than one sequence of tags or more than one parse per sentence. This is justified in cases where it is too unreliable to assign just one best parse. A commonly used measure is the percentage of sentences for which there is the correct structure among the n best hypotheses of the parser, e.g., $n = 5$ or $n = 20$. We use a different notion in order to exploit the probabilities that are assigned by a statistical tagger or parser.

Our tagger calculates the best sequence of tags given a sequence of words and all alternative sequences that have probabilities "close to" the probability of the best sequence. Closeness is defined by a threshold on the quotient of probabilities (cf. section 4.3). The rationale behind the assignment of more than one tag is to pick out the unreliable cases (for which the accuracy is not expected to be high) and emit two or more tags in order to increase the chance that the correct tag is at least among these tags. But, of course, we would like to assign just one tag to most of the words in order to reduce ambiguity.

When using a threshold on the quotient of probabilities, we are interested in the following measures:

Given a threshold θ on the quotient of the best and the alternative probability,

- how often is the correct tag among the proposed tags?
- what is the average number of tags proposed per word?

Given the quotient of the best and the second best probability,

- what is the accuracy of the best assignment?

Chapter 7

Evaluation Corpora and Results

Chapter Summary

This chapter reports on the evaluation of the proposed components of a partial parsing and annotation system, i.e, part-of-speech tagging, tagging grammatical functions, assigning phrase categories, interactive annotation and partial parsing. Evaluation is performed by using four corpora, covering two languages and two domains. All methods can be successfully applied to the four corpora, requiring only some portion of annotated data for training.

The presented methods are evaluated for corpora in two different languages and from two different domains. The languages are German and English. The first domain is newspaper text, the second domain is transliterated appointment dialogues. We use four corpora for the four different combinations of languages and domains (cf. table 7.1).

To perform the experiments, a parameterizable statistical tagger, TnT, was implemented (Brants, 1996b). Note that tagging grammatical functions, assigning phrase categories and Cascaded Markov Models are extensions or generalizations of part-of-speech tagging. For lattice tagging, the extension TnTL was implemented (Brants, 1999b).

The next section gives an overview and short descriptions of these corpora. Then, we present results for part-of-speech tagging, assigning grammatical functions, assigning phrase categories, interactive annotation, and partial parsing for these four corpora.

Table 7.1: Use of four corpora to cover two languages and two domains.[1]

	newspaper text	transliterated dialogues
German	NEGRA Corpus Frankfurter Rundschau 280,000 tokens	Verbmobil Corpus appointment dialogues 120,000 tokens
English	Penn Treebank Wall Street Journal 1,200,000 tokens	Verbmobil Corpus appointment dialogues 150,000 tokens

7.1 Corpora

The experiments are performed on four different corpora for two languages and two domains (see table 7.1). The corpora are described in the following sections.

7.1.1 NEGRA Corpus

The German NEGRA corpus consists of newspaper texts (Frankfurter Rundschau) that are annotated with predicate-argument structures (Skut et al., 1997). It was developed in the project NEGRA (Nebenläufige grammatische Verarbeitung; Concurrent Grammar Processing) at the Saarland University, Saarbrücken. Part of it was part-of-speech tagged at the IMS Stuttgart. The annotation consists of four parts: 1) a non-projective predicate-argument structure, 2) phrase categories (NP, PP, ...) that are annotated as node labels, 3) grammatical functions (subject, direct object, pre-nominal genitive, ...) that are annotated as edge labels, and 4) part-of-speech tags. Non-projective parts were converted to context-free structures before structural parsing experiments were performed (see Skut, Brants, Krenn, & Uszkoreit, 1997, for details of this conversion). The labeling tasks of assigning grammatical functions and phrase categories are the same for projective and non-projective parts. The corpus is still growing. By the time the experiments were performed, October 1998, it had a size of approx. 16,000 sentences (280,000 tokens).

[1]Sizes in the table are as of October 1998. The NEGRA and Verbmobil corpora are still growing.

7.1.2 Penn Treebank

We use the Wall Street Journal as contained in the Penn Treebank for our experiments. The annotation consists of four parts: 1) a context-free structure augmented with traces to mark movement and discontinuous constituents, 2) phrase categories that are annotated as node labels, 3) a small set of grammatical functions that are annotated as extensions to the node labels, and 4) part-of-speech tags (Marcus et al., 1993). Opposed to the NEGRA corpus, only a fraction of all grammatical functions is marked. The Wall Street Journal part of the Penn Treebank consists of approx. 50,000 sentences (1.2 million tokens).

7.1.3 Verbmobil Corpus, German and English Parts

The Verbmobil Corpora consist of transliterated spoken appointment dialogues. The Verbmobil project collects data for German, English, and Japanese. We use those parts of the German and English data that are syntactically annotated (Stegmann & Hinrichs, 1998). The annotations also consist of four parts: 1) a context-free structure, 2) phrase categories that are annotated as node labels, 3) grammatical functions that are annotated as edge labels, and 4) part-of-speech tags. As of October 1998, the size of the German part is approx. 120,000 (15,000 sentences), the size of the English part is around 150,000 tokens (12,000 sentences).

7.2 Part-of-Speech Tagging

We evaluate the tagger's performance under several aspects. First of all, we determine the tagging accuracy averaged over ten iterations. The overall accuracy, as well as separate accuracies for known and unknown words are measured. The tagger is rather language- and tagset-independent. It can be trained for virtually any language that delimits words with white space and for which sufficient training material is available. If additional material is available, e.g., large amounts of untagged data, a manually created lexicon or morphological component, or hand-crafted disambiguation rules, the performance of the tagger can be significantly improved. We evaluate the effect of these methods for the NEGRA corpus.

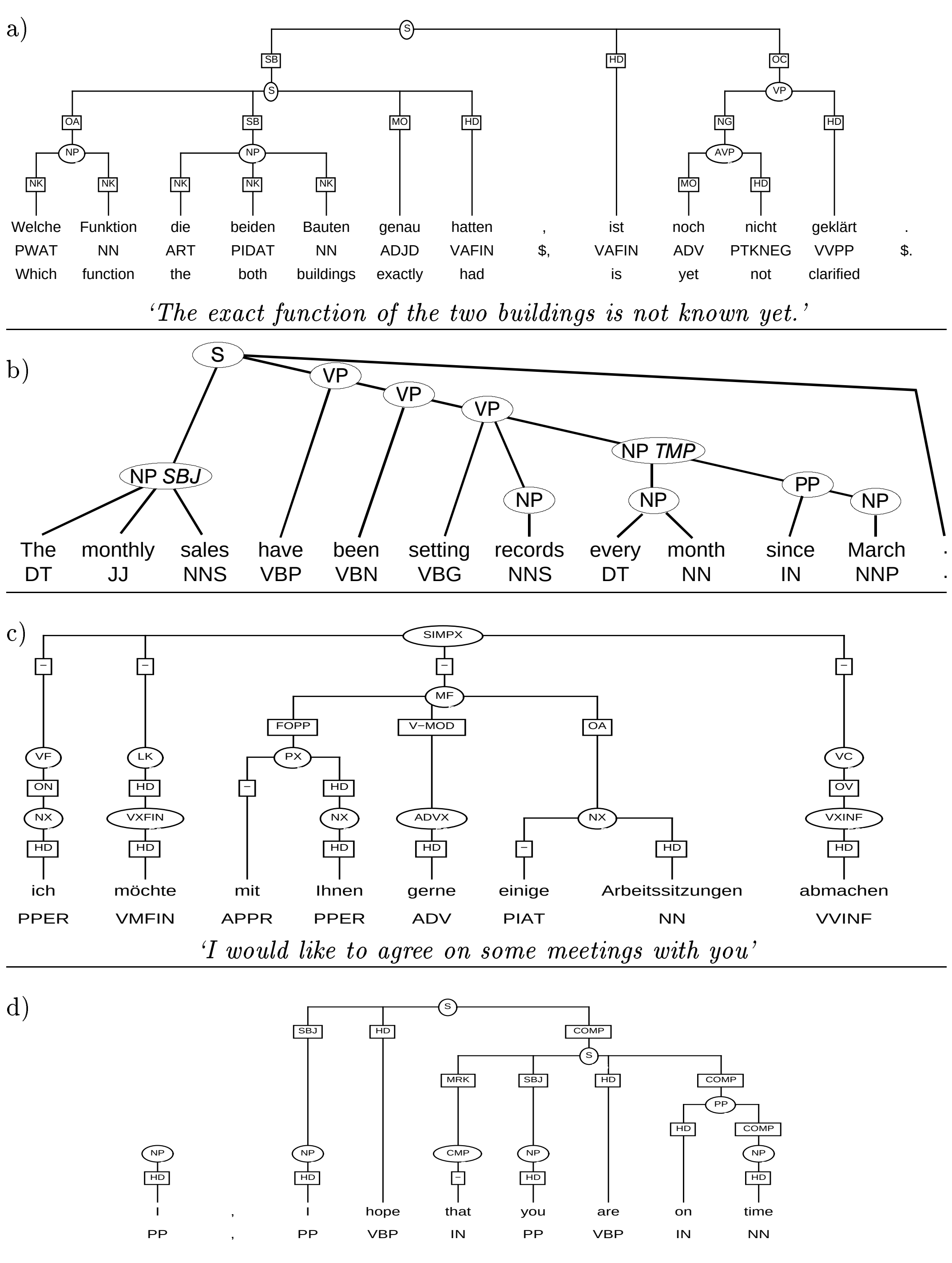

a)
S
SB HD OC
S VP
OA SB MO HD NG HD
NP NP AVP
NK NK NK NK NK MO HD
Welche Funktion die beiden Bauten genau hatten , ist noch nicht geklärt .
PWAT NN ART PIDAT NN ADJD VAFIN $, VAFIN ADV PTKNEG VVPP $.
Which function the both buildings exactly had is yet not clarified
'The exact function of the two buildings is not known yet.'
b)
S
VP VP VP
NP SBJ
NP TMP
NP NP PP NP
The monthly sales have been setting records every month since March .
DT JJ NNS VBP VBN VBG NNS DT NN IN NNP .
c)
SIMPX
– – – –
MF
FOPP V-MOD OA
VF LK PX VC
ON HD – HD OV
NX VXFIN NX ADVX NX VXINF
HD HD HD HD – HD HD
ich möchte mit Ihnen gerne einige Arbeitssitzungen abmachen
PPER VMFIN APPR PPER ADV PIAT NN VVINF
'I would like to agree on some meetings with you'
d)
S
SBJ HD COMP
S
MRK SBJ HD COMP
PP
HD COMP
NP NP CMP NP NP
HD HD – HD HD
I , I hope that you are on time
PP , PP VBP IN PP VBP IN NN

Figure 7.1: Example sentences taken from the four corpora that are used in the evaluation: a) NEGRA corpus, b) Penn Treebank, c) Verbmobil German, d) Verbmobil English.

Second, learning curves are presented that indicate the performance when using training corpora of different sizes, starting with as few as 1,000 tokens and ranging to the size of the entire corpus (minus the test set).

An important characteristic of statistical taggers is that they not only assign tags to words but also probabilities in order to rank different assignments. The third set of experiments investigates alternative assignments that are "close to" the best assignment, with "close to" referring to the distance of the respective probabilities.

The fourth series of experiments measures the reliability of the tagger: the tagger estimates the quality of a particular tag assignment in order to allow statements like "it is relatively safe to assign this particular tag" or "we cannot decide whether tag A or tag B should be assigned".

7.2.1 Part-of-Speech Tagging Accuracy

The tagging accuracy is the percentage of correctly assigned tags. We distinguish the overall accuracy, taking into account all tokens in the test corpus, and the accuracy for known and unknown tokens, taking into account the correct assignments for known or unknown tokens, only. The latter two are interesting since usually unknown tokens are much more difficult to process than known tokens, for which a list of valid tags can be found in the lexicon.

Tagging accuracies for the four corpora are shown in table 7.2. Accuracy for tagging the Penn Treebank is at least on a par with results reported elsewhere in the literature for single systems (e.g., compared to the best results presented in Ratnaparkhi, 1996)[2]. Results for the other corpora cannot be compared directly to results of most other investigations due to differences in the domain or tagset. But since we used very hard criteria (no additional restrictions on the test data, test data is guaranteed to be unseen during training, tests were repeated 10 times with different partitions) the results can be judged as very good.

All corpora have in common that results for known words are much better than for unknown tokens. In both domains, results for the English corpus are better than those for the German corpus. In case of the newspaper domain, this can be

[2]In fact, our percentages are higher, but the difference of 0.1 is statistically not significant.

explained by the larger number of unknown tokens in German text since separate results for known and unknwon words are very close together.

For both languages, new texts in the newspaper domain have a higher rate of unknown (i.e., previously unseen) words than in the appointments domain. The difference is very large for German. There are 13.1% unknown tokens when testing the NEGRA corpus, but only 1.7% in the Verbmobil corpus.

Accuracy of unknown words in the Verbmobil corpus has a very large standard deviation (5.91 and 5.39) compared to the NEGRA corpus and the Penn Treebank (1.35 and 0.54). We do not have an explanation for this difference.

We used additional information and processing steps in order to increase the accuracy for the NEGRA corpus. Results for the improved model are shown in table 7.3. Model 2 uses a large untagged corpus (approx. 40 million tokens) to re-estimate lexical probabilities of unknown words. We used one iteration of the expectation-maximization procedure to generate lexical probabilities for unknown words (transitional frequencies and lexical frequencies for known words are unchanged). This yields an improvement of 0.3%.

7.2.2 Learning Curves for Part-of-Speech Tagging

The accuracy of a tagger heavily depends on the amount of available training data. The larger the corpus, the better the coverage of the lexicon generated from the corpus, and the better are the probability estimates. We expect better tagging results for larger training corpora.

This section presents the learning curves of the tagger, i.e., the accuracy depending on the amount of training data. The curves for the four corpora are shown in figures 7.2 to 7.5. Training length is the number of tokens used for training. Each training length was tested ten times, training and test sets were disjoint, results are averaged. The training length is given on a logarithmic scale.

It is remarkable that tagging accuracy for known words is very high even for very small training corpora. This means that we have a good chance of getting the right tag if a word is seen at least once during training. Accuracy for known words even has a local minimum in the Penn Treebank for medium training sizes around

Table 7.2: Part-of-speech tagging accuracy for four different corpora. The table shows the percentage of unknown tokens, separate accuracies and standard deviations for known and unknown tokens, as well as the overall accuracy.

corpus	percentage	known		unknown		overall	
	unknowns	acc.	σ	acc.	σ	acc.	σ
NEGRA corpus	13.1%	97.7%	0.28	86.6%	1.01	96.3%	0.27
Penn Treebank	2.8%	97.1%	0.12	84.2%	0.54	96.7%	0.13
VM German	1.7%	97.3%	0.44	77.2%	5.91	96.9%	0.52
VM English	1.2%	97.9%	0.52	78.0%	5.39	97.7%	0.59

Table 7.3: Tagging results for the NEGRA corpus using an extended model.

		known	unknown	overall
		(86.9%)	(13.1%)	(100%)
1.	Base model:	97.7%	86.6%	96.3%
2.	Base + unknown word re-estimation:	97.7%	88.8%	96.6%

Table 7.4: Increase in accuracy when doubling the training size from half of the corpus size to the full size (minus test set). This is used to roughly estimate the gain in accuracy when further increasing the training size.

Corpus	from	to	Δaccuracy
NEGRA corpus	130,000	260,000	+0.80
Penn Treebank	500,000	1,000,000	+0.09
Verbmobil (German)	50,000	100,000	+0.55
Verbmobil (English)	70,000	140,000	+0.15

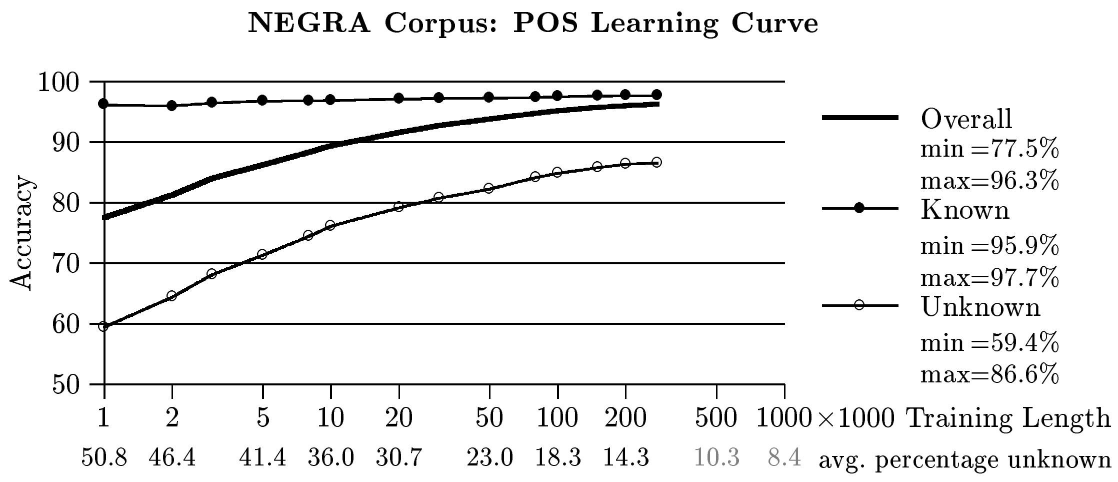


Figure 7.2: Learning curve for tagging the NEGRA corpus. The training sets of variable sizes as well as test sets of 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged. Percentages of unknowns for 500k and 1000k training are determined from an untagged extension.

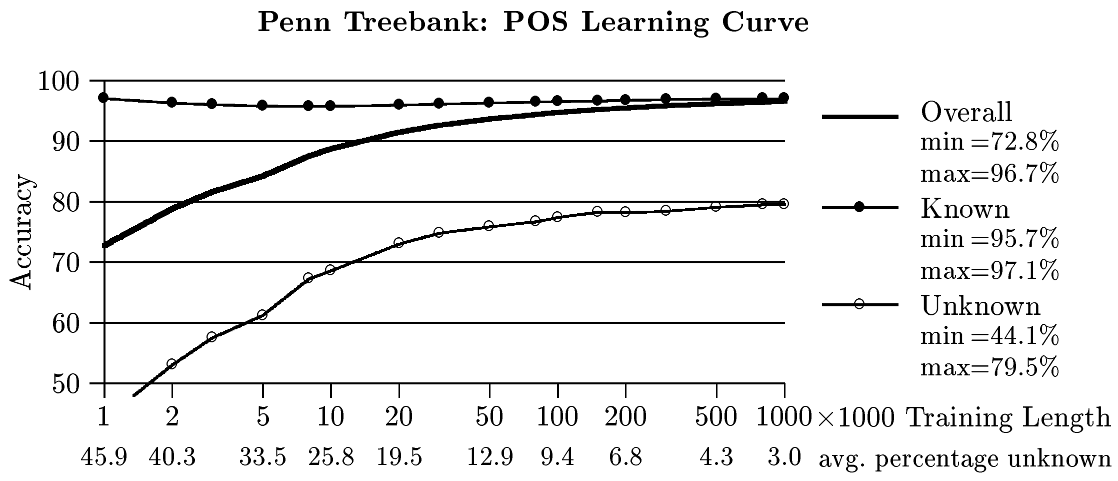


Figure 7.3: Learning curve for tagging the Penn Treebank. The training sets of variable sizes as well as test sets of 100,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

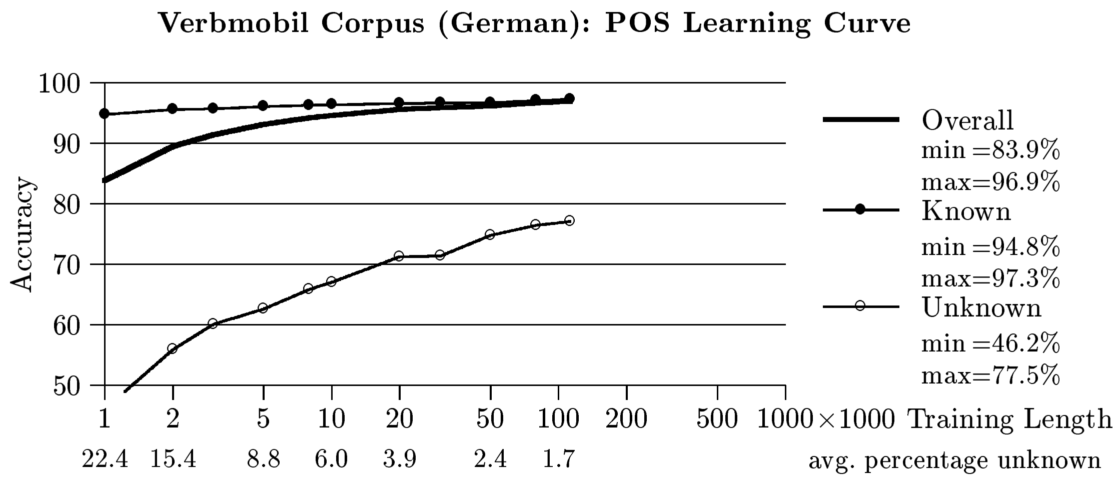


Figure 7.4: Learning curve for tagging the Verbmobil corpus (German). The training sets of variable sizes as well as test sets of 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

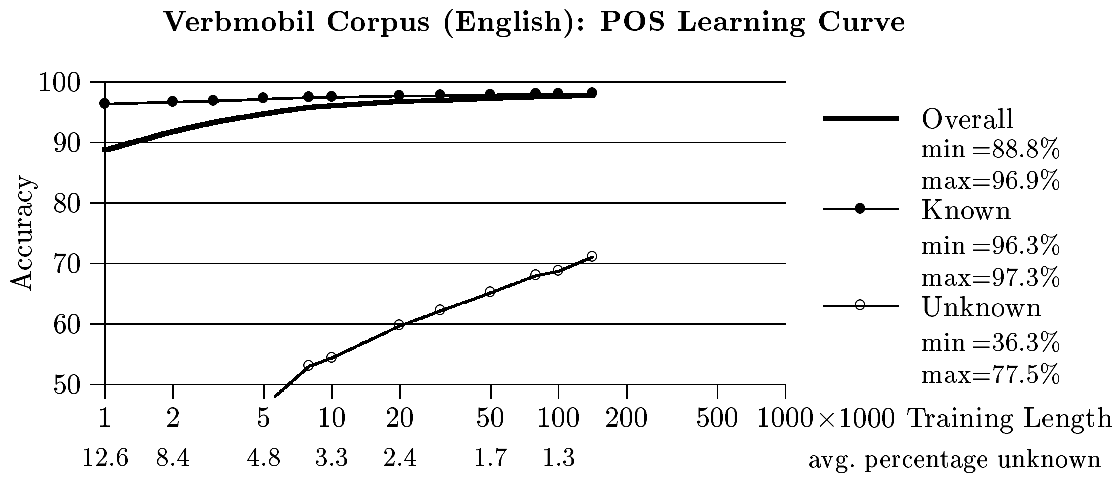


Figure 7.5: Learning curve for tagging the Verbmobil corpus (English). The training sets of variable sizes as well as test sets of 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

10,000 tokens (training 1,000: accuracy 97.1; training 10,000: accuracy 95.7). We repeated the ten-fold crossvalidation several times with randomly chosen sets of partitions. The repetitions confirmed that the local minimum is a stable effect. Our hypothesis is that this is an effect of the increasing ambiguity rate for known words when increasing the size of the training corpus which makes disambiguation more difficult. But the other corpora do not show this effect.

Average percentages of unknown tokens is shown in the bottom line of each diagram. As expected, the percentage is larger for German than for English for both the written and spoken data. The percentage of unknown words is lower for the transliterated spoken data, which reflects the restricted domain.

Overall accuracy for all corpora is still growing even when using the complete corpus (minus the test set) for training. So we can expect a further increase in accuracy when extending the training corpora, even though it may be a small increase. Table 7.4 shows the increase in accuracy at the end points of the curves when doubling the training size. Expected accuracy gains for the next doubling of the training size is smaller than that shown in the table.

The most room for improvement is in handling unknown words. But only test sets of the NEGRA corpus have a large amount of unknown tokens (13.1% for the largest training size). This percentage is small for the other three corpora (between 1.2 and 2.8%), so the effect of improved unknown word handling on the overall accuracy is expected to be small.

7.2.3 Remaining Ambiguity for Part-of-Speech Tagging

We exploit the fact that the tagger not only determines tags, but also assigns probabilities. If there is an alternative that has a probability "close to" that of the best assignment, this alternative can be viewed as almost equally well suited. The following series of experiments investigates the possibility of assigning more than one tag to a token if there is a close alternative candidate, thereby leaving some ambiguity in the tagger's output (cf. section 4.3). The notion of "close to" is expressed by the distance of probabilities, and this in turn is expressed by the quotient of probabilities. So, the distance of the probabilities of a best tag t_{best} and an alternative tag

t_{alt} is expressed by $p(t_{best})/p(t_{alt})$, which is some value greater or equal to 1 since the best tag assignment has the highest probability.

Figures 7.6 to 7.9 show the *recall* and *conditional accuracies* when taking more and more alternatives into account. For *recall*, an assignment is counted as correct if either of the alternatives is correct. The curves start at distance factor 1, i.e., only the best tag[3] is assigned. Note that this is (almost) the standard notion of accuracy, and the percentages at this point are the same as the averages in section 7.2.1 (Tagging Accuracy) and the curves' end points in section 7.2.2 (Learning Curves).

As expected, accuracy increases when allowing a greater ambiguity rate in the output. Accuracy gain is largest at the beginning of the curves. Adding all tags within beam $\beta = 2$ increases the accuracy by about 1% for all corpora, thereby leaving between 1.02 and 1.03 tags per word in the corpus (i.e., 2 or 3 out of 100 words get more than one tag).

The existence of alternative tag assignments and their probabilities can be used as an indicator for the reliability of the best tag assignment. The *conditional accuracy* shows the accuracy of all words for which exactly one tag is assigned. The accuracy of the complement set shows the accuracy of the best tag of all words for which more than one tag is assigned.

We see from the curves that the greater the difference between the best and the other tags, the higher the accuracy of the best tag. For a subset of words we can determine the correct tag with very high accuracy. Whether a word belongs to this subset can be determined based on the tags' probabilities.

It is interesting to see that maximum accuracy is achieved for medium quotients of the best and the second best tags (between $\beta = 100$ and $\beta = 1000$), and that the conditional accuracy falls slightly for larger β. This occurs for all four corpora. We think this effect is due to missing tags in the initial set of tags proposed by the tagger for a particular word. For known words, the situation occurs if the correct tag is not among those in the lexicon. For unknown words, the situation occurs if the unknown word handler proposes a set of tags for the word, but the correct one

[3]Additionally, alternative tags with identical probabilities are assigned for distance factor 1, but this almost never occurs.

Table 7.5: 5 most frequent part-of-speech tagging errors for the NEGRA corpus (total 284,190 tokens).

	correct		assigned			rel.	abs.
	tag	freq_c	tag	freq_a(% of freq_c)		error	error
1.	NE	15069	NN	2092	(13.9%)	19.6%	0.74%
2.	VVFIN	11595	VVINF	667	(5.8%)	6.3%	0.23%
3.	NN	58563	NE	615	(1.1%)	5.8%	0.22%
4.	VVFIN	11595	VVPP	425	(3.7%)	4.0%	0.15%
5.	ADJA	16843	NN	270	(1.6%)	2.5%	0.10%

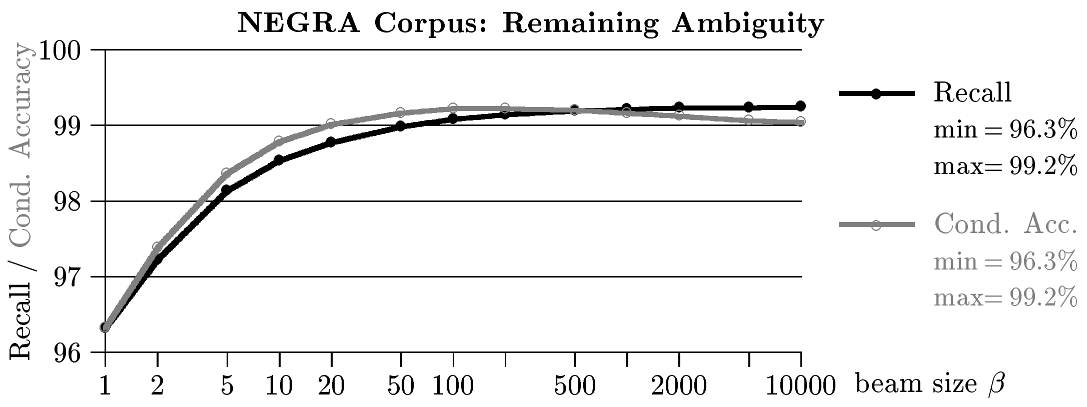
is not in the set. In this situation, the tagger may be very confident in its choice *given the set of hypothetical tags.*

7.2.4 Most Frequent Tagging Errors

We performed a tagging error analysis for the NEGRA corpus. Table 7.5 shows the 5 most frequent errors with respect to the number of tokens tagged incorrectly. It shows the correct tag, its frequency in the corpus, the wrongly assigned tag, the frequency of the mis-assignment, the percentage in relation to the frequency of the correct tag, its relative contribution to the error rate (all errors = 100%), and its absolute contribution to the error rate (all errors = 3.7%).

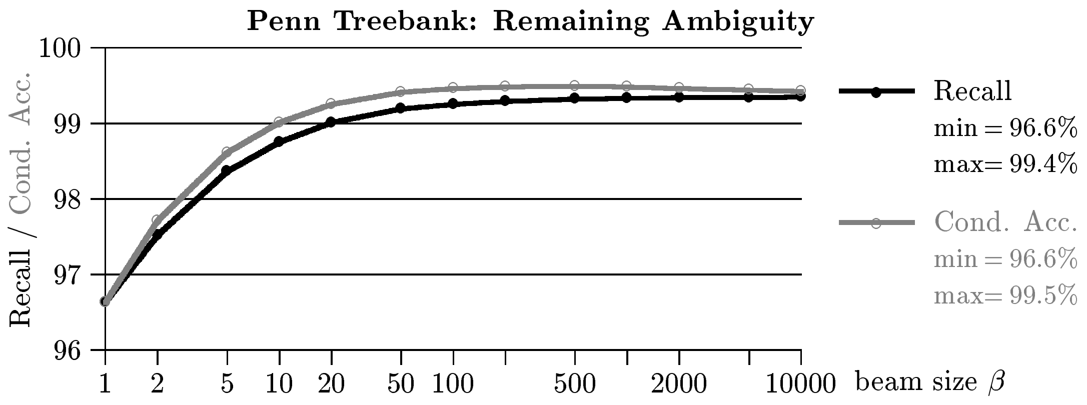
Most errors stem from the confusion of proper nouns (NE) and common nouns (NN). These are very hard to distinguish for a tagger. The same is true for the distinction of finite verbs (VVFIN) and non-finite verbs (VVINF and VVPP). This can be explained by the fact that often the disambiguation context for a verb is the presence or absence of a finite auxiliary, and the distance of the auxiliary and full verb is larger than the tagger's window.

At rank 5, we find adjectives (ADJA) that are mis-tagged as common noun (NN). This is surprising at first sight, since German common nouns are capitalized while adjectives are not, and the tagger takes capitalization into account. But a closer look at the data reveals that mis-tagged adjectives most often occur at the beginning of a sentence (which means they are capitalized) or consist of adjectives not starting with a letter (e.g., *17jährige* – 17-year-old).



beam size β	1	2	5	10	20	50	100		500		2000		10000
acc. of complement	–	51.7	61.9	68.5	74.1	79.7	82.8	85.4	88.1	89.6	90.7	91.8	92.1
avg. # tags/token	1.00	1.03	1.07	1.11	1.16	1.25	1.34	1.45	1.63	1.79	1.97	2.21	2.36

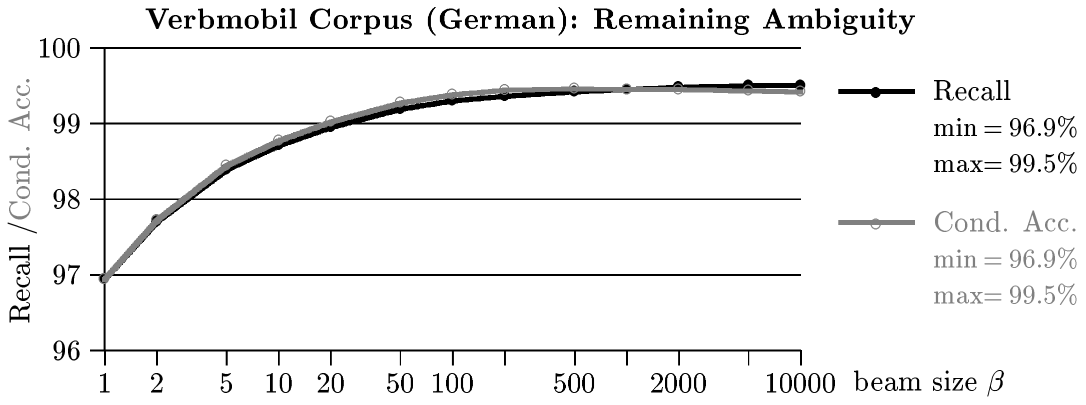
Figure 7.6: Tagging accuracy for the NEGRA corpus when some ambiguity remains after tagging (see below for a description).



beam size β	1	2	5	10	20	50	100		500		2000		10000
acc. of complement	–	53.5	62.8	68.9	73.9	79.3	82.6	85.2	87.5	88.8	89.8	91.0	91.6
avg. # tags/token	1.00	1.02	1.06	1.09	1.12	1.17	1.21	1.26	1.32	1.37	1.42	1.50	1.55

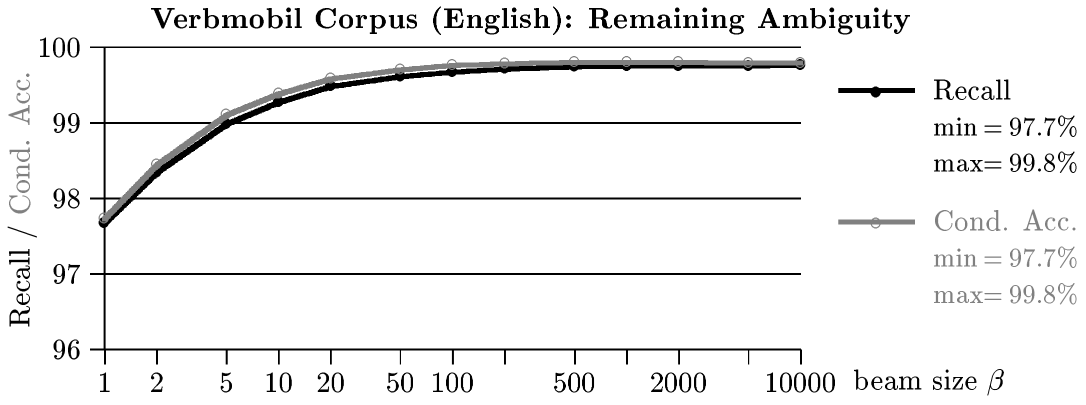
Figure 7.7: Tagging accuracy for the Penn Treebank when some ambiguity remains after tagging.

The best tag t_{best} and all tags t_{alt} with probabilities within the beam β (having $p(t_{best})/p(t_{alt}) \leq \beta$) are assigned. Recall is the number of words for which the correct tag is among the assigned tags. Conditional accuracy is the accuracy for those words that are assigned exactly one tag. Accuracy of complement is the accuracy of the best tag for those words that are assigned more than one tag. Additionally, we give the average number of assigned tags per token.



beam size β	1	2	5	10	20	50	100		500		2000		10000
accuracy of complement	–	61.7	71.2	77.3	81.2	85.0	87.5	89.5	91.4	92.4	93.2	93.9	94.2
avg. # tags/token	1.00	1.02	1.06	1.09	1.13	1.19	1.25	1.32	1.42	1.50	1.58	1.68	1.75

Figure 7.8: Tagging accuracy for the Verbmobil corpus (German part) when some ambiguity remains after tagging (see below for a description).



beam size β	1	2	5	10	20	50	100		500		2000		10000
acc. of complement	–	52.9	66.5	74.1	78.7	83.0	85.1	86.9	88.9	90.0	90.9	91.8	92.3
avg. # tags/token	1.00	1.02	1.05	1.08	1.11	1.16	1.19	1.22	1.27	1.31	1.35	1.39	1.42

Figure 7.9: Tagging accuracy for the Verbmobil corpus (English part) when some ambiguity remains after tagging.

The best tag t_{best} and all tags t_{alt} with probabilities within the beam β (having $p(t_{best})/p(t_{alt}) \leq \beta$) are assigned. Recall is the number of words for which the correct tag is among the assigned tags. Conditional accuracy is the accuracy for those words that are assigned exactly one tag. Accuracy of complement is the accuracy of the best tag for those words that are assigned more than one tag. Additionally, we give the average number of assigned tags per token.

The tagger would benefit most from an improved handling of these errors.

7.2.5 Summary of Part-of-Speech Tagging Results

Average part-of-speech tagging accuracy is between 96% and 98%, depending on language, tagset, and size of the training corpus. These results are at least on a par with state-of-the-art results found in the literature. Accuracy for known tokens is significantly higher than for unknown tokens. For the German newspaper data, results are 11% better when the word was seen before and therefore is in the lexicon, than when it was not seen before (97.7% vs. 86.6%). Accuracy for known tokens is high even with very small amounts of training data. As few as 1,000 tokens are sufficient to achieve 95%–96% accuracy for them. It is important for the tagger to have seen a word at least once. From this we conclude that a tagger strongly benefits from an additional, possibly manually created lexicon that handles words which did not occur during the training phrase.

The base technique of the tagger only uses an annotated corpus, no additional resources. Learning tags for unknown words from a large un-tagged corpus increases tagging accuracy for the NEGRA corpus by about 0.3%.

Stochastic taggers assign probabilities to tags. We exploit the probabilities to leave ambiguity after tagging if the probability of the second best assignment is close to that of the best assignment. This identifies reliable and unreliable assignments and we leave some ambiguity in the output of the tagger if the assignment of a unique tag would be unreliable. As a surprising result, we find that reliability does not monotonically increase with distance of first and second best assignment. Instead, there is a maximum accuracy for "medium" distances. This effect may be explained by missing tags in the lexicon.

7.3 Tagging Grammatical Functions

The task investigated in this section consists of the assignment of a grammatical function to each element within each phrase. As input, the categories of the children and the category of the parent node are given. The technique is presented in section 4.4. Here, we assume that the lower layer is assigned correctly. This assumption is justified in the annotation task, when a human annotator creates structures bottom up.

We evaluate the tagger's performance under several aspects. First of all, we determine the tagging accuracy averaged over ten iterations. As can be seen from the results, this accuracy is heavily dependent on the information encoded within the states of the Markov Models that are used for tagging.

Second, learning curves for assigning grammatical functions are generated to study the influence of the amount of training data.

In a third series of experiments, the tagger not only assigns the highest ranked grammatical function, but also alternatives with probabilities that are close to the probability of the best function.

7.3.1 Accuracy of Assigning Grammatical Functions

The tagging accuracy is the percentage of correctly assigned grammatical functions. In contrast to part-of-speech tagging, there is no need to distinguish known and unknown tokens. We assign functions to tags. The number of tags is small (around 50 in the corpora used), therefore almost all of the tags are seen in the training corpus.

Tagging accuracies are presented in table 7.6. Results range from 95.2% (Verbmobil German) to 97.3% (Verbmobil English). Results for the Penn Treebank are not directly comparable to the results for the other three corpora because most grammatical functions in the Penn Treebank are not explicitly marked. So the tagger assigns the label "no function" most of the time, and those labels that are explicitly marked are difficult to recognize. The accuracy for explicitly marked grammatical functions in the Penn Treebank is only 71%.

We experimented with different encodings for the NEGRA corpus (see table 7.7). The base model encodes grammatical functions as states of a Markov Model (cf. section 4.4.2). This is not optimal. Additionally encoding tags in the states significantly improves results ("extended tags" in table 7.7). We observed that a lot of errors occured because of lacking morphological information. Adding this information with the help of a morphological analyzer (cf. section 4.4.3) significantly improves results for the base model as well as for the extended model.

7.3.2 Learning Curves for Grammatical Functions

The accuracy of a tagger heavily depends on the amount of training data available. Usually, the larger the training corpus, the better the parameter estimates and the better the tagging results.

This section presents the learning curves of the tagger for grammatical functions, i.e., the accuracy depending on the amount of training data. The curves for the four corpora are shown in figures 7.10 to 7.13. Training length is the number of tokens at the word level used for training. As an example, the sentence in figure 7.1a on page 104 consists of 13 tokens, which would be given as training size, and 16 grammatical functions. Each training length was tested ten times, training and test sets were disjoint, results are averaged. The training length is given on a logarithmic scale.

An accuracy of 90% can be reached with very little training data: around 2,000 tokens in the NEGRA corpus, around 1,000 tokens in the other corpora. But the slopes of the curves rapidly decrease. Overall accuracy for all corpora is still growing even when using the complete corpus (minus the test set) for training. So we can expect a further increase in accuracy when extending the training corpora, even though it may be a small increase. Table 7.8 shows the increase in accuracy at the end points of the curves when doubling the training size. Since the slopes of the curves are decreasing, we expect that accuracy gains for the next doubling of the training size are smaller than those shown in the table. The largest gain is expected for the NEGRA corpus.

Table 7.6: Accuracy and standard deviation for assigning grammatical functions using four different corpora. The corpora were divided into 90% training and 10% test set. Experiments were repeated 10 times, results were averaged.

corpus	acc.	σ
NEGRA corpus	96.3%	0.40
Penn Treebank	96.1%	0.11
Verbmobil German	95.2%	0.29
Verbmobil English	97.3%	0.48

Table 7.7: Accuracy and standard deviation for assigning grammatical functions in the NEGRA corpus, using four different models (see text).

model	acc.	σ
1) Base model	94.7%	0.45
2) Base + Morphology	95.3%	0.46
3) Extended model	95.6%	0.43
4) Extended + Morphology	96.3%	0.40

Table 7.8: Increase in accuracy of assigning grammatical functions when doubling the training size from half of the corpus size to the full size (minus test set). This is used to roughly estimate the gain in accuracy when further increasing the training size.

Corpus	from	to	Δaccuracy
NEGRA corpus	130,000	260,000	0.32
Penn Treebank	500,000	1,000,000	0.04
Verbmobil (German)	50,000	100,000	0.07
Verbmobil (English)	70,000	140,000	0.02

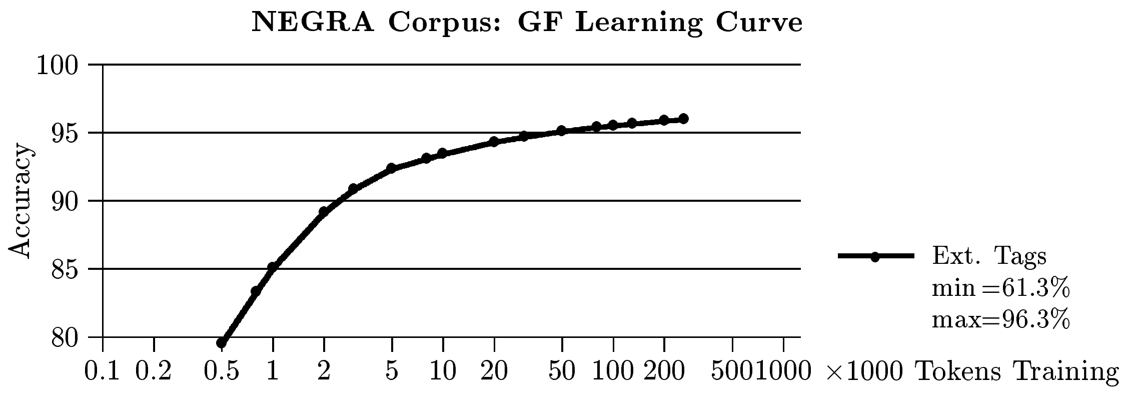


Figure 7.10: Learning curves for assigning grammatical functions in the NEGRA corpus. The training sets of variable sizes as well as test sets of around 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

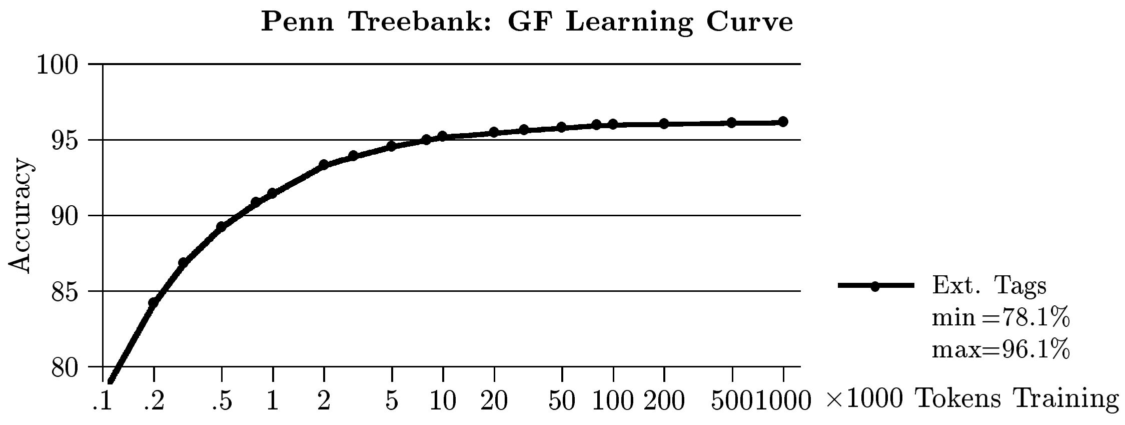


Figure 7.11: Learning curves for assigning grammatical functions in the Penn Treebank. The training sets of variable sizes as well as test sets of around 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

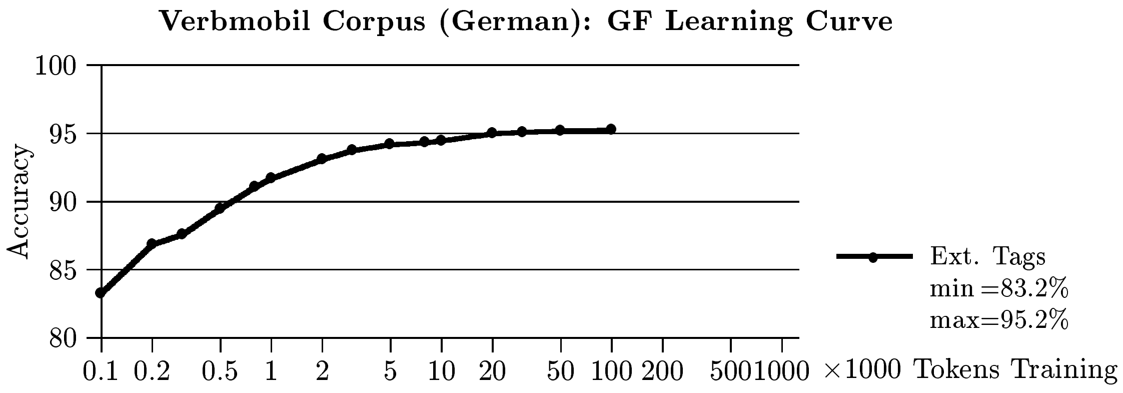


Figure 7.12: Learning curves for assigning grammatical functions in the Verbmobil corpus (German part). The training sets of variable sizes as well as test sets of around 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

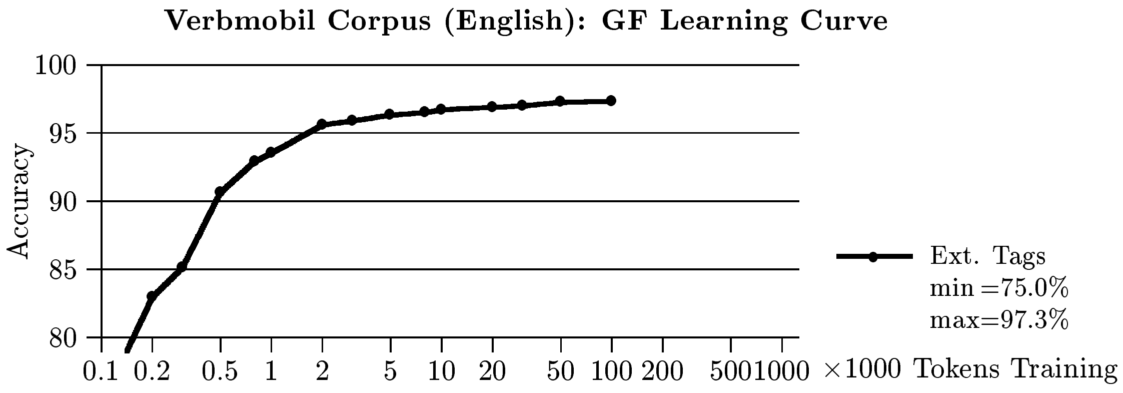


Figure 7.13: Learning curves for assigning grammatical functions in the Verbmobil corpus (English part). The training sets of variable sizes as well as test sets of around 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

7.3.3 Remaining Ambiguity for Grammatical Functions

We exploit the fact that the tagger not only determines tags, but also assigns probabilities. If there is an alternative that has a probability “close to” that of the best assignment, this alternative can be viewed as almost equally well suited. The following series of experiments investigates the possibility of assigning more than one tag to a token if there is a close alternative candidate, thereby leaving some ambiguity in the tagger’s output (cf. section 4.3). The notion of “close to” is expressed by the distance of probabilities, and this in turn is expressed by the quotient of probabilities. So, the distance of the probabilities of a best tag t_{best} and an alternative tag t_{alt} is expressed by $p(t_{best})/p(t_{alt})$, which is some value greater or equal to 1 since the best tag assignment has the highest probability.

Figures 7.14 to 7.17 show the *recall* and *conditional accuracies* when taking more and more alternatives into account. For *recall*, an assignment is counted as correct if either of the alternatives is correct. The curves start at beam factor 1, i.e., only the best tag (or alternative tags with identical probabilities) are assigned. Note that this is (almost) the standard notion of accuracy, and the percentages at this point are the same as the averages in section 7.3.1 (accuracy) and the curves’ end points in section 7.3.2 (learning).

We see that gains in accuracy are large for small beams and additional gains become smaller for large beams. Accuracy comes close to 100% for the largest beam that was tested ($\beta = 10000$). This is different to the part-of-speech tagging result. There, the asymptote was well below 100% because of lexical errors, i.e., the word is found in the lexicon but does not have the correct tag. Lexical errors for tagging grammatical functions are negligible.

Leaving selected ambiguity significantly increases accuracy, but the price is that the output is not fully disambiguated. The bottom line in each of the four figures shows the average number of tags that are assigned per token. An average of 1.06 means that we assign 106 functions to 100 tags. An actual application needs to make a compromise between ambiguity rate of the output and expected accuracy.

The existence of alternative tag assignments and their probabilities can be used as an indicator for the reliability of the best tag assignment. The *conditional accuracy*

Table 7.9: 5 most frequent errors when assigning grammatical functions in the NEGRA corpus (total 337,887 functions).

	correct		assigned			rel.	abs.
	tag	freq_c	tag	freq_a	(% of freq_c)	error	error
1.	OA	4136	SB	917	(22.2%)	7.4%	0.27%
2.	SB	19484	OA	802	(4.1%)	6.5%	0.24%
3.	PD	726	OC	602	(82.9%)	4.9%	0.18%
4.	MO	1726	OC	508	(29.4%)	4.1%	0.15%
5.	PD	2848	MO	500	(17.6%)	4.0%	0.15%

shows the accuracy of all words for which exactly one tag is assigned. The accuracy of the complement set shows the accuracy of the best tag of all words for which more than one tag is assigned.

We see from the curves that the greater the difference between the best and the other tags, the higher the accuracy of the best tag. For a subset of words we can determine the correct tag with very high accuracy. Whether a word belongs to this subset can be determined based on the tags' probabilities.

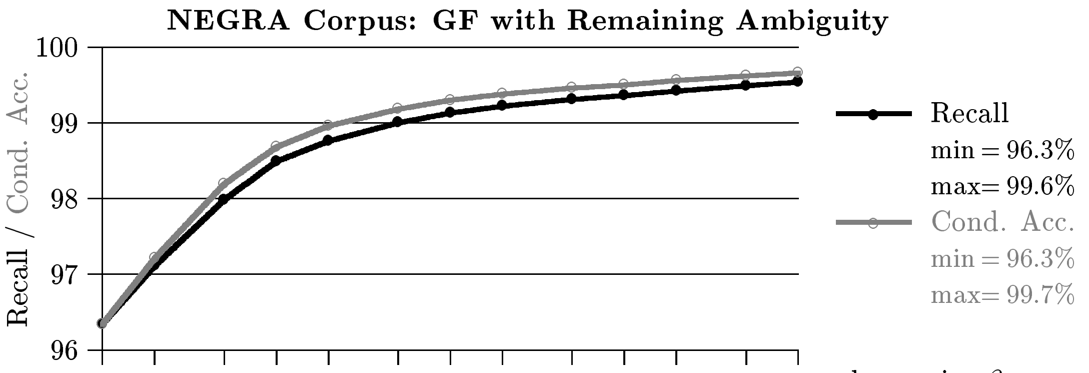
7.3.4 Most Frequent Assignment Errors

We performed an error analysis for the NEGRA corpus. Table 7.9 shows the 5 most frequent errors with respect to the number of tokens tagged incorrectly[4]. It shows the correct grammatical function, its frequency in the corpus, the wrongly assigned function, the frequency of the mis-assignment, the percentage in relation to the frequency of the correct tag, its relative contribution to the error rate (all errors = 100%), and its absolute contribution to the error rate (all errors = 3.7%).

In contrast to part-of-speech tagging, we do not find one high-frequent error (mis-tagging NE as NN accounted for 19.6% of all errors). The most frequent error in assigning grammatical functions is the wrong assignment of SB (subject) when it should be OA (direct object).

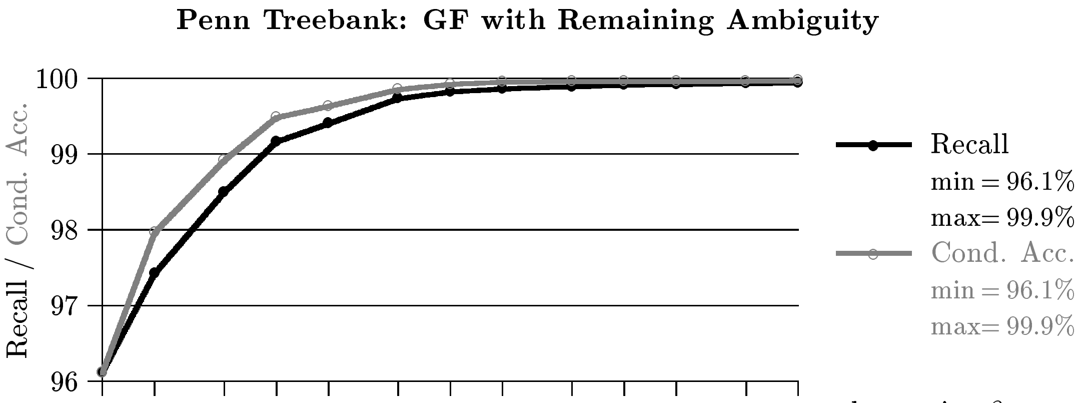
In the error table, we find that several functions are very hard to process. We

[4]We used the extended model and encoding of morphological information to obtain these frequencies.



1	2	5	10	20	50	100		500		2000		10000	beam size β
–	53.1	63.2	69.8	73.5	78.6	81.5	83.6	85.7	86.5	87.2	87.7	88.2	acc. of complement
1.00	1.02	1.06	1.10	1.14	1.19	1.24	1.29	1.37	1.42	1.46	1.51	1.54	avg. # tags/token

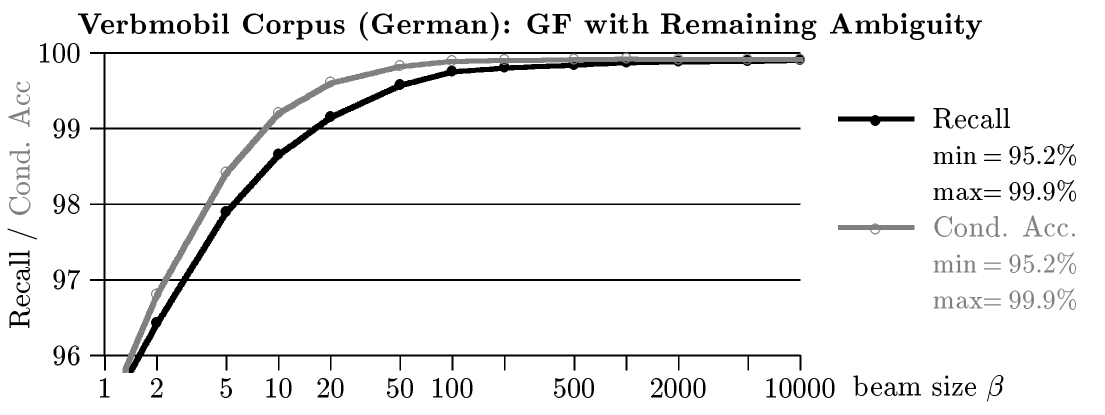
Figure 7.14: Accuracy for assigning grammatical functions in the NEGRA corpus when some ambiguity remains after tagging. See below for a description.



1	2	5	10	20	50	100		500		2000		10000	beam size β
–	40.1	49.0	60.6	64.7	75.6	77.9	79.4	79.9	81.0	82.7	86.4	86.7	acc. of complement
1.00	1.02	1.06	1.10	1.14	1.19	1.24	1.29	1.37	1.42	1.46	1.51	1.54	avg. # tags/token

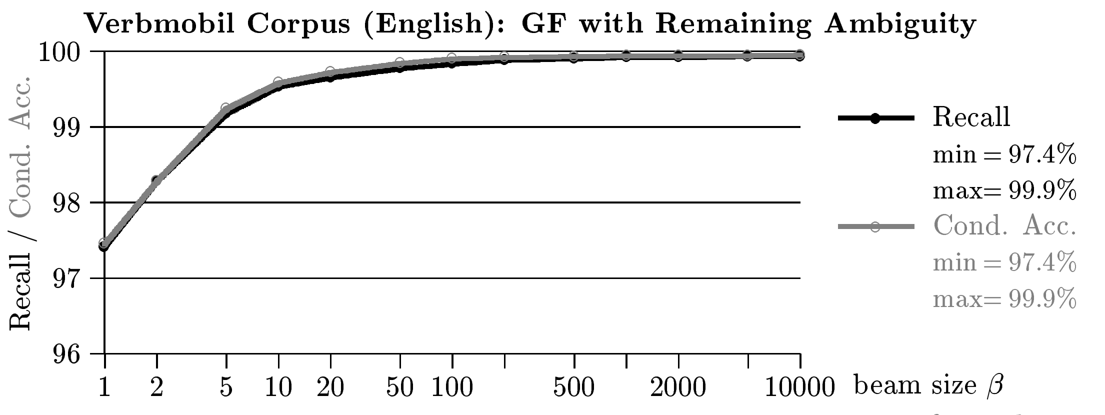
Figure 7.15: Accuracy for assigning grammatical functions in the Penn Treebank when some ambiguity remains after tagging.

The best tag t_{best} and all tags t_{alt} with probabilities within the beam β (having $p(t_{best})/p(t_{alt}) \leq \beta$) are assigned. Recall is the number of words for which the correct tag is among the assigned tags. Conditional accuracy is the accuracy for those words that are assigned exactly one tag. Accuracy of complement is the accuracy of the best tag for those words that are assigned more than one tag. Additionally, we give the average number of assigned tags per token.



beam size β	1	2	5	10	20	50	100		500		2000		10000
acc. of complement	–	46.6	54.8	66.9	71.7	77.6	78.8	79.5	81.0	83.5	89.0	91.0	91.8
avg. # tags/token	1.00	1.04	1.12	1.20	1.30	1.48	1.59	1.71	1.85	2.00	2.24	2.47	2.58

Figure 7.16: Accuracy for assigning grammatical functions in the Verbmobil corpus (German part) when some ambiguity remains after tagging. See below for a description.



beam size β	1	2	5	10	20	50	100		500		2000		10000
acc. of complement	–	56.4	65.1	68.8	72.2	78.5	83.2	85.0	86.8	87.4	90.1	91.1	91.6
avg. # tags/token	1.00	1.02	1.06	1.08	1.10	1.15	1.21	1.27	1.33	1.38	1.48	1.54	1.60

Figure 7.17: Accuracy for assigning grammatical functions in the Verbmobil corpus (English part) when some ambiguity remains after tagging.

The best tag t_{best} and all tags t_{alt} with probabilities within the beam β (having $p(t_{best})/p(t_{alt}) \leq \beta$) are assigned. Recall is the number of words for which the correct tag is among the assigned tags. Conditional accuracy is the accuracy for those words that are assigned exactly one tag. Accuracy of complement is the accuracy of the best tag for those words that are assigned more than one tag. Additionally, we give the average number of assigned tags per token.

see that 82.9% of all predicatives (PD) are mis-tagged as clausal object (OC). The methods presented here use the category of the parent node, the category of the child node, and the functions in the context as information sources. As the results show, this is not sufficient to recognize the function PD. The functions OA (direct object) and MO (modifier) also have very low recall rates.

All functions with high error rates belong to the sentence and VP level. In contrast, the functions at the NP and PP level have low error rates, none of them is among the top 5.

7.3.5 Summary of Results for Assigning Grammatical Functions

Average accuracy for assigning grammatical functions ranges from 95% to 97%, depending on language, tagset, and size of the training corpus. Opposed to part-of-speech tagging, there is no need for handling unknown words, because functions are assigned to tags, and the number of tags is small. Usually all tags are seen in the training corpus, except for very small training corpora.

Accuracy also depends on the encoding of information in the Markov Model. A different schema for constructing the states of the model yields an increase of around 1% for the NEGRA corpus. Accuracy can be further increased by using morphological information. We found results to be 0.7% better when using the ambiguous output of a morphological analyzer for this corpus.

The learning curves show that we can reach 90% accuracy with as little as 1,000 words of training data. The slope of the learning curves is still positive when using the complete corpus (minus test set) for training, but expected further increase is very small after 100k words training.

By exploiting the probabilities that are assigned by the tagger we can select alternative tags that are left in the output. This increases the chance that the correct grammatical function is in the set of assigned functions. The quotient of the best and second best probability is a direct indicator for the expected accuracy of the best function that is assigned. Practical applications will select a threshold in the quotient that marks the limit up to which alternative functions are assigned.

7.4 Assigning Phrase Categories

The task investigated in this section consists of the assignment of phrase categories to each nonterminal node of a structure. As input, the categories of the children are given. The technique is presented in section 4.5. Here, we assume that the lower layer is assigned correctly. This assumption is justified in the annotation task, when a human annotator creates structures bottom up.

We evaluate the tagger's performance under several aspects. First of all, we determine the tagging accuracy averaged over ten iterations. Second, learning curves for assigning grammatical functions are generated to study the influence of the amount of training data. In a third series of experiments, the tagger not only assigns the highest ranked grammatical function, but also alternatives with probabilities that are close to the probability of the best function.

7.4.1 Accuracy of Assigning Phrase Categories

The tagging accuracy is the percentage of correctly assigned phrase categories. In contrast to part-of-speech tagging, there is no need to distinguish known and unknown tokens. We assign categories based on the children's tags. The number of tags is small, therefore almost all of the tags are seen in the training corpus.

Tagging accuracies are presented in table 7.10. Results range from 92.1% (Verbmobil German) to 97.4% (NEGRA Corpus). The German Verbmobil Corpus has a very high standard deviation, i.e., results for the 10 iterations varied very much. The reason for this variation is not clear.

We tested if the (possibly ambiguous) output of a morphological analyzer improves the assignment of phrase categories (cf. section 4.4.3). Table 7.11 shows accuracy for models without and with using morphology. Results are significantly better when using morphology, although the difference is smaller than for the assignment of grammatical functions.

7.4.2 Learning Curves for Phrase Categories

The accuracy of a tagger heavily depends on the amount of available training data. Usually, the larger the training corpus, the better the parameter estimates and the

Table 7.10: Accuracy and standard deviation for assigning phrase categories using four different corpora. The corpora were divided into 90% training and 10% test set. Experiments were repeated 10 times, results were averaged.

corpus	acc.	σ
NEGRA corpus	97.4%	0.44
Penn Treebank	95.8%	0.13
Verbmobil German	92.1%	1.37
Verbmobil English	92.8%	0.56

Table 7.11: Accuracy and standard deviation for assigning phrase categories with and without using a morphological analyzer for the NEGRA corpus.

model	acc.	σ
1) without morphology	97.0%	0.42
2) with morphology	97.4%	0.44

Table 7.12: Increase in accuracy of assigning phrase categories when doubling the training size from half of the corpus size to the full size (minus test set). This is used to roughly estimate the gain in accuracy when further increasing the training size.

Corpus	from	to	Δaccuracy
NEGRA corpus	130,000	260,000	0.27
Penn Treebank	500,000	1,000,000	0.02
Verbmobil (German)	50,000	100,000	0.02
Verbmobil (English)	70,000	140,000	0.01

better the tagging results.

This section presents the learning curves of the tagger for grammatical functions, i.e., the accuracy depending on the amount of training data. The curves for the four corpora are shown in figures 7.18 to 7.21. Training length is the number of tokens at the word level used for training. As an example, the sentence in figure 7.1a consists of 13 tokens, which would be given as training size, and 6 non-terminal nodes. Each training length was tested ten times, training and test sets were disjoint, results are averaged. The training length is given on a logarithmic scale.

An accuracy of 90% can be reached with very little training data: around 2,000 tokens in the NEGRA corpus, only around 500 tokens in the other corpora. But the slopes of the curves rapidly decrease. Overall accuracy for all corpora is still growing when using the complete corpus (minus the test set) for training, even though the increase is very small except for the NEGRA corpus. For this corpus, we can expect a further increase in accuracy when extending the training corpora. Table 7.12 shows the increase in accuracy at the end points of the curves when doubling the training size. Since the slopes of the curves are decreasing, we expect accuracy gains for the next doubling of the training size are smaller than those shown in the table.

7.4.3 Remaining Ambiguity for Phrase Categories

We exploit the fact that the tagger not only determines categories, but also assigns probabilities. If there is an alternative that has a probability "close to" that of the best assignment, this alternative can be viewed as almost equally well suited. The following series of experiments investigates the possibility of assigning more than one category to a phrase if there is a close alternative candidate, thereby leaving some ambiguity in the tagger's output (cf. section 4.3). The notion of "close to" is expressed by the distance of probabilities, and this in turn is expressed by the quotient of probabilities. So, the distance of the probabilities of a best tag t_{best} and an alternative tag t_{alt} is expressed by $p(t_{best})/p(t_{alt})$, which is some value greater or equal to 1 since the best tag assignment has the highest probability.

Figures 7.22 to 7.25 show the *recall* and *conditional accuracies* when taking more and more alternatives into account. For *recall*, an assignment is counted as correct

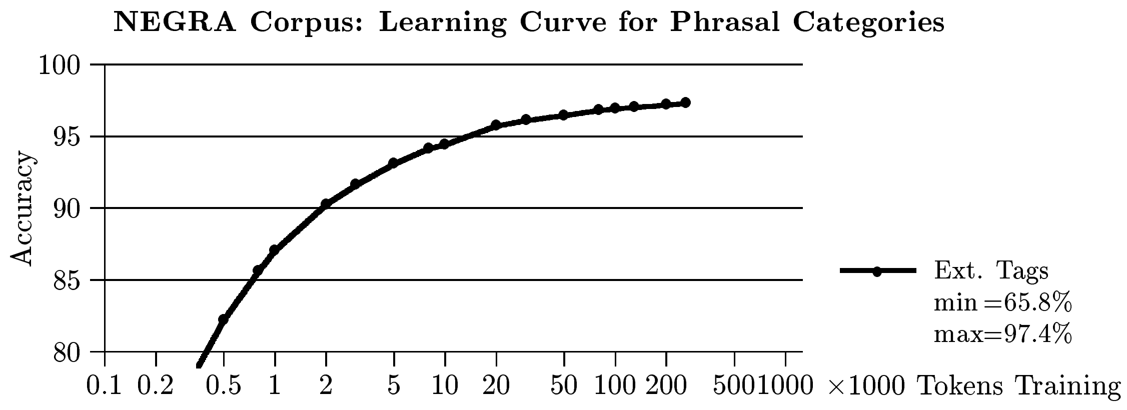


Figure 7.18: Learning curves for assigning phrase categories in the NEGRA corpus. The training sets of variable sizes as well as test sets of around 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

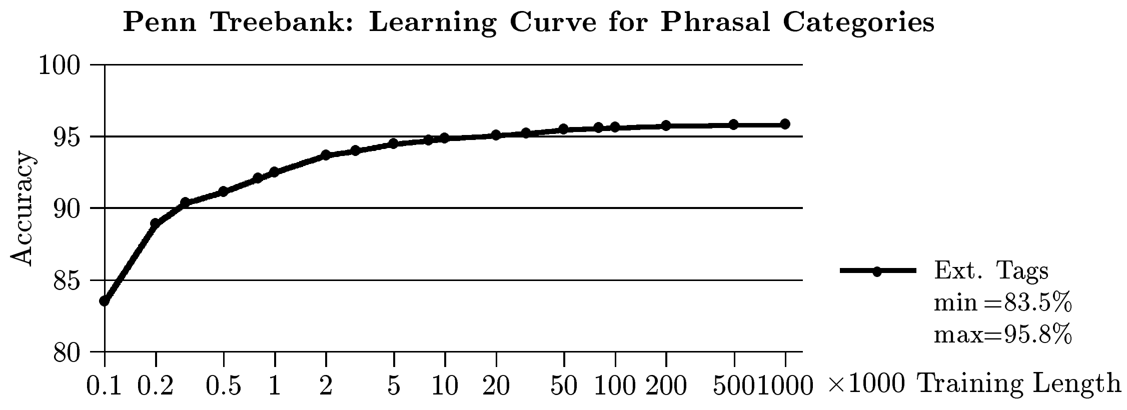


Figure 7.19: Learning curves for assigning phrase categories in the Penn Treebank. The training sets of variable sizes as well as test sets of around 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

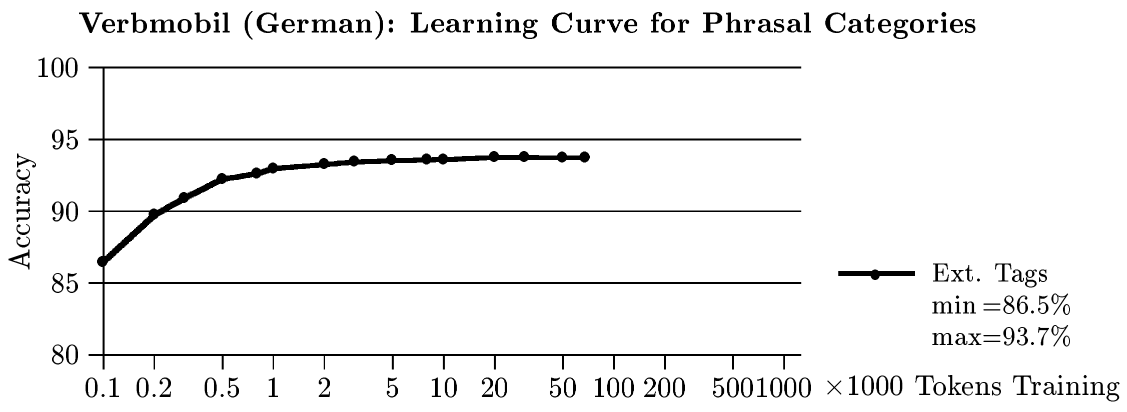


Figure 7.20: Learning curves for assigning phrase categories in the Verbmobil Corpus (German Part). The training sets of variable sizes as well as test sets of around 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

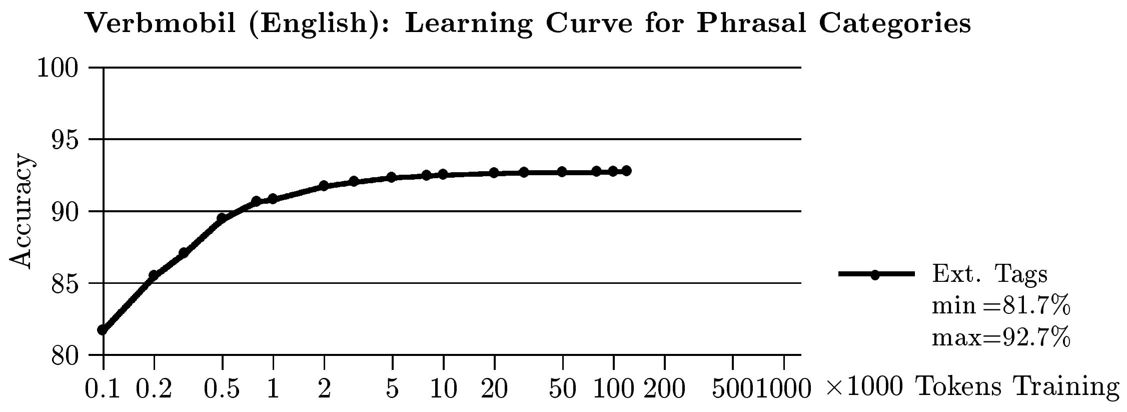


Figure 7.21: Learning curves for assigning phrase categories in the Verbmobil Corpus (English Part). The training sets of variable sizes as well as test sets of around 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

if either of the alternatives is correct. The curves start at beam factor 1, i.e., only the best tag or alternative tags with identical probabilities are assigned. Note that this is (almost) the standard notion of accuracy, and the percentages at this point are the same as the averages in section 7.4.1 (accuracy) and the curves' end points in section 7.4.2 (learning).

We again see that gains in accuracy are large for small beams and additional gains become smaller for larger beams. Accuracy comes close to 100% for the largest beam that was tested ($\beta = 10000$).

Leaving selected ambiguity significantly increases accuracy, but the price is that the output is not fully disambiguated. The bottom line in each of the four figures shows the average number of tags that are assigned per token. An average of 1.06 means that we assign 106 functions to 100 tags. An actual application needs to make a compromise between ambiguity rate of the output and expected accuracy.

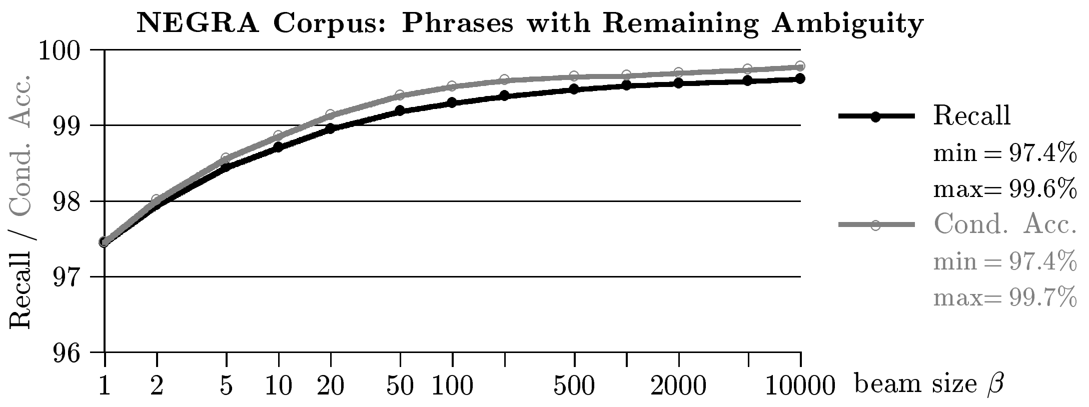
The existence of alternative tag assignments and their probabilities can be used as an indicator for the reliability of the best tag assignment. The *conditional accuracy* shows the accuracy of all words for which exactly one tag is assigned. The accuracy of the complement set shows the accuracy of the best tag of all words for which more than one tag is assigned.

We see from the curves that the greater the difference between the best and the other tags, the higher the accuracy of the best tag. For a subset of words we can determine the correct tag with very high accuracy. Whether a word belongs to this subset can be determined based on the tags' probabilities.

7.4.4 Most Frequent Assignment Errors

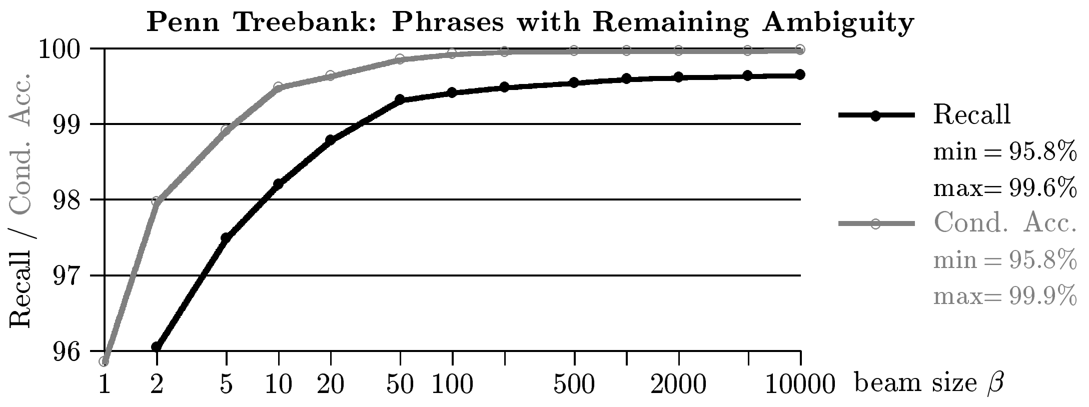
We performed an error analysis for the NEGRA corpus. Table 7.13 shows the 5 most frequent errors with respect to the number of phrases assigned incorrectly. It shows the correct phrase category, its frequency in the corpus, the wrongly assigned function, the frequency of the mis-assignment, the percentage in relation to the frequency of the correct tag, its relative contribution to the error rate (all errors = 100%), and its absolute contribution to the error rate (all errors = 2.6%).

The most frequent error is the wrong assignment of NP to an S phrase. This is



beam size β	1	2	5	10	20	50	100		500		2000		10000
acc. of complement	–	51.8	66.8	72.7	78.4	85.9	87.8	89.2	93.6	94.7	95.1	95.4	95.6
avg. # tags/token	1.00	1.01	1.04	1.06	1.10	1.19	1.25	1.33	1.61	1.75	1.85	1.93	1.99

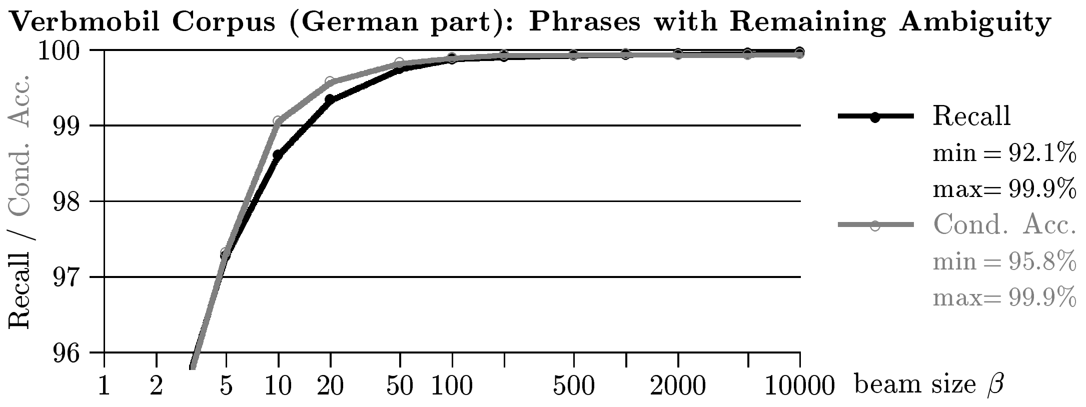
Figure 7.22: Tagging accuracy for assigning phrase categories in the NEGRA corpus, leaving selected ambiguities in the output. See below for a description.



beam size β	1	2	5	10	20	50	100		500		2000		10000
acc. of complement	–	40.1	49.0	60.6	64.7	75.6	77.9	79.4	79.9	81.0	82.7	86.4	86.7
avg. # tags/token	1.00	1.07	1.18	1.45	1.48	1.51	1.72	1.93	1.97	2.01	2.13	2.39	2.39

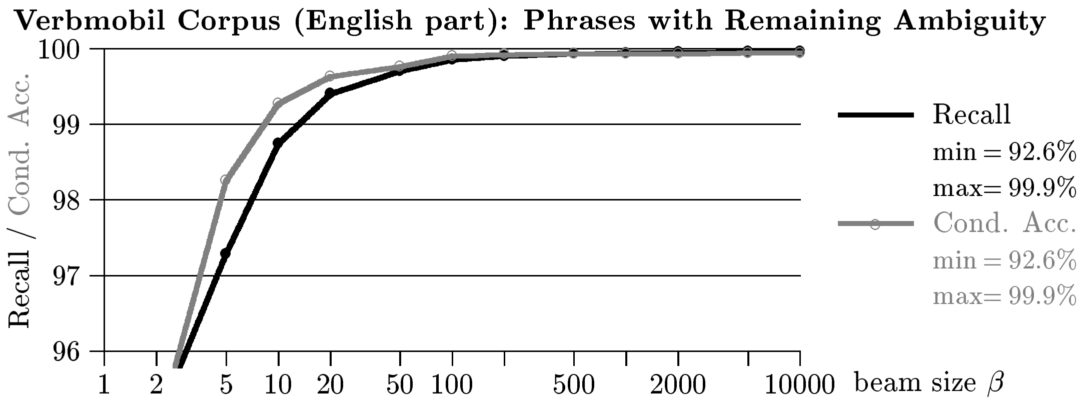
Figure 7.23: Tagging accuracy for assigning phrase categories in the Penn Treebank, leaving selected ambiguities in the output.

The best tag t_{best} and all tags t_{alt} with probabilities within the beam β (having $p(t_{best})/p(t_{alt}) \leq \beta$) are assigned. Recall is the number of words for which the correct tag is among the assigned tags. Conditional accuracy is the accuracy for those words that are assigned exactly one tag. Accuracy of complement is the accuracy of the best tag for those words that are assigned more than one tag. Additionally, we give the average number of assigned tags per token.



beam size β	1	2		5	10	20	50	100		500		2000		10000
acc. of complement	–	49.7	57.8	63.0	73.7	75.4	77.3	78.1	78.7	81.5	84.4	87.6	88.4
avg. # tags/token	1.00	1.05	1.22	1.34	1.52	1.77	1.96	2.02	2.19	2.34	2.55	2.92	3.00

Figure 7.24: Tagging accuracy for assigning phrase categories in the Verbmobil corpus (German part), leaving selected ambiguities in the output (see below for a description).



beam size β	1	2		5	10	20	50	100		500		2000		10000
acc. of complement	–	61.0	65.0	72.5	74.9	77.3	83.0	83.5	83.9	84.6	85.7	86.0	86.2
avg. # tags/token	1.00	1.07	1.18	1.41	1.48	1.53	1.72	1.93	2.05	2.19	2.28	2.49	2.50

Figure 7.25: Tagging accuracy for assigning phrase categories in the Verbmobil corpus (English part), leaving selected ambiguities in the output.

The best tag t_{best} and all tags t_{alt} with probabilities within the beam β (having $p(t_{best})/p(t_{alt}) \leq \beta$) are assigned. Recall is the number of words for which the correct tag is among the assigned tags. Conditional accuracy is the accuracy for those words that are assigned exactly one tag. Accuracy of complement is the accuracy of the best tag for those words that are assigned more than one tag. Additionally, we give the average number of assigned tags per token.

Table 7.13: 5 most frequent errors when assigning phrase categories in the NEGRA corpus (total 114782 phrases).

	correct		assigned			rel.	abs.
	tag	freq_c	tag	freq_a	(% of freq_c)	error	error
1.	S	23007	NP	241	(1.1%)	8.2%	0.21%
2.	NP	33556	S	195	(0.6%)	6.6%	0.17%
3.	DL	282	CS	169	(59.9%)	5.7%	0.14%
4.	VP	10098	S	147	(1.5%)	5.0%	0.13%
5.	NP	33556	PP	132	(0.4%)	4.5%	0.12%

surprising at first sight. Looking at the data, we find that these are either elliptical sentences or incomplete sentences in headlines, e.g.:

So kletterten Daimler um knapp 35 Mark, (S *Siemens* (PP *um fast 25*)) *, ...*

[Daimler rose by almost 35 Mark, (S Siemens (PP by almost 25)) , ...]

(S *Nachtragsetat* (PP *auf der Tagesordnung*))

[(S Additional budget (PP on the agenda))]

Both S nodes are erroneously tagged as NP. The error also occurs for the opposite direction, e.g., in

(NP *wer* (AVP *auch immer*))

[whoever]

the NP is parsed as an elliptical S.

The third most frequent error is interesting because it identifies a type of constituent that is very hard to recognize. 59.9% of all DL nodes (discourse level constituent) are recognized as CS (coordinated sentence). The former are mainly sentences containing direct speech with an introducing sentence like "Peter said ...", which are syntactically very hard to distinguish from two coordinated sentences.

The distinction of S and VP in the NEGRA annotation scheme is that S nodes immediately dominate finite verbs while VP nodes immediately dominate non-finite verbs. The confusion of these categories is mainly due to elliptical VPs without the verb.

The fifth error again is surprising at first sight, since usually PP and NP can be distinguished easily be the presence or absence of a preposition. But according to the NEGRA schema there is a special case where this is not possible. *Als* (as/than) is part of an NP if used with comparatives *größer als Peter* (taller than Peter) and part of a PP if not used with comparatives *Peter als Lehrer* (Peter as teacher), which accounts for most of the NP/PP errors.

7.4.5 Summary of Results for Assigning Phrase Categories

Average accuracy for assigning phrase categories ranges from 92% to over 97%, depending on language, tagset, and size of the training corpus. In contrast to part-of-speech tagging, there is no need for handling unknown words, because phrase categories are assigned to tags, and the number of tags is small. So usually all tags are seen in the training corpus, except for very small training corpora.

Accuracy also depends on the encoding of information in the Markov Model. Adding (possibly ambiguous) morphological information to the phrase categories yields an increase of around 0.4% for the NEGRA corpus.

The learning curves show that we can reach an accuracy of 90% with as little as 500 words of training data for the Penn Treebank and the Verbmobil corpora; 2,000 words are needed for the NEGRA corpus.

By exploting the probabilities that are assigned by the tagger we can select alternative tags that are left in the output. This increases the chance that the correct phrase category is in the set of assigned functions. The quotient of the best and second best probability is a direct indicator for the expected accuracy of the best function that is assigned.

7.5 Cascaded Markov Models – Interactive Mode

This section presents results for applying Cascaded Markov Models layer by layer, phrase by phrase. Each time a new phrase is suggested by the automatic process, it is manually corrected before proceeding to the next phrase. This is the *annotation mode* as described in section 5.1.1 and used in combination with a graphical user interface for efficient interactive annotation.

In order to determine the number of layers necessary for processing, we extracted the depths of structures for sentences in the corpora. Table 7.14 shows the average depths of annotations. We additionally show the depth that is necessary to cover 99% of all sentences (i.e., for 99% of the sentences the number of layers is smaller than or equal to this number), and the maximum depth. Our intention is to cover 99% of all sentences. As we see from the table, this requires between 8 and 20 layers, depending on the corpus. The difference in the number of layers is probably mainly due to the different annotation schemes.

In the following series of experiments, we will use the number of Markov Models that is necessary to cover 99% of all sentences for each corpus.

7.5.1 Node Results

In the annotation mode, a partial annotation is given, and the task is to add a new node (phrase). Each time a node is suggested, it is either confirmed or declined by an annotator, so the given partial annotation can always be assumed to be correct. This section presents results on the accuracies of suggested phrases.

We do not have accuracy values for actual human-computer interactions. Instead, we use existing corpus data and simulate the annotation process in the following way.

Annotation starts without any structural element. One new phrase is suggested by the Markov Models for layer 1. This is compared to the correct structure and counted as either a correct or incorrect node. If the suggestion is correct, the process continues by suggesting a next phrase. If it is incorrect, it is corrected to form the nearest matching correct phrase. The nearest matching phrase is the one that needs fewest additions or deletions of constituents.[5] Thus, after each step either the correctly suggested phrase or the nearest match of an incorrect suggestion is added to the annotation. The procedure continues until the annotation is complete, and we count the percentage of correct suggestions.

The automatic process has the option not to make any suggestion. In this case, the next phrase that is to be inserted is randomly chosen from the correct annotations.

The simulation is only an approximation of the annotation process. The human annotator can use more complex operations to create the annotation. But it yields rough estimaties of the accuracy of the automatic part in interactive annotation.

Table 7.15 shows accuracy results for the interactive annotation task. "Phrase suggested" lists the percentage of cases in which the system made a suggestion, "phrase correct" is the percentage of correct phrases (of all suggested phrases), "phrase and label" is the labeled accuracy. We also show the percentage of "almost correct" suggestions. These are those phrases that can be converted to correct phrases by removing or adding at most one constituent (e.g., missing or spurious adverbs, PPs, etc.). Although the suggested phrase is not entirely correct, it is nevertheless helpful for the annotator since only a small change creates a correct phrase.

The annotator has the possibility to reject a phrase. In this case, probabilities are re-computed and the next-best phrase is suggested. We think that the automation is useful if either the first or second suggestion is correct. For efficiency reasons, the annotator probably better creates the structure manually if the correct suggestion is further down in the line. Therefore, we present accuracies for the first or second suggestion in table 7.16.

In order to measure the effect on human annotation speed, we measured the time needed to annotate sentences with and without the automatic generation of structures for the NEGRA corpus. Labeling (i.e., parts-of-speech, grammatical functions, and phrase categories) is automated in both cases[6]. Average annotation time when

[5]If there is more than one nearest matching phrase, one of them is chosen randomly.

[6]We do not have representative figures for annotation times when both labeling and structures are fully manually generated. This mode was used in the very beginning of the project when

Table 7.14: Depths of structures. The table shows the number of layers of annotations in the four corpora (excluding the part-of-speech layer). It shows the average number, the number of layers that is necessary to cover 99% of the sentences, and the maximum depth found in the corpus.

	number of layers		
corpus	avg.	99%	max.
NEGRA corpus	4.0	9	15
Penn Treebank	9.1	20	35
VM German	3.8	8	14
VM English	4.8	10	16

Table 7.15: Percentage of partial annotations in which a new phrase was suggested, and accuracies for suggested phrases. +/- 1 means that the suggested phrase either misses at most one child constituent or has at most one spurious constituent. The corpora were divided into 90% training and 10% test set. Experiments were repeated 10 times, results were averaged.

corpus	phrase suggested	phrase correct	phrase and label correct	phrase +/- 1	phrase and label +/- 1
NEGRA corpus	99.8%	69.8%	67.8%	87.1%	82.9%
Penn Treebank	99.9%	77.0%	75.4%	85.9%	81.0%
Verbmobil German	99.9%	91.9%	88.7%	96.6%	92.0%
Verbmobil English	99.9%	82.9%	71.8%	94.1%	80.6%

Table 7.16: Percentage of partial annotations in which a new phrase was suggested, and accuracies for suggested phrases, taking into account the first and second suggestions (i.e., the annotator has to reject at most one phrase until being presented a correct phrase). +/- 1 means that the suggested phrase misses at most one child constituent or has at most one spurious constituent. The corpora were divided into 90% training and 10% test set. Experiments were repeated 10 times, results were averaged.

first or second suggestion	phrase suggested	phrase correct	phrase and label correct	phrase +/- 1	phrase and label +/- 1
NEGRA corpus	99.8%	83.6%	81.1%	94.5%	90.6%
Penn Treebank	99.9%	91.0%	87.6%	95.7%	90.3%
Verbmobil German	99.9%	97.4%	93.6%	99.2%	94.8%
Verbmobil English	99.9%	91.1%	77.9%	97.9%	83.6%

Table 7.17: Annotation times in seconds for one sentence. The times are averaged over 2,000 sentences (avg. length 17.5 tokens/sentence) and two trained annotators.

labeling	structure	avg. time	std. deviation	avg. tokens/hour
automated	manual	71s	65	900
automated	automated	50s	61	1,300

using the graphical user interface as presented in section 5.1.4 and the automatic labeling of nodes and edges as presented in sections 4.4 and 4.5, is 71 seconds per sentence. Average annotation time when additionally using the interactive generation of structures is 50 seconds. This is a reduction of annotation time by approx. 30%. The times where averaged over 2,000 sentences for two trained annotators. Average sentence length was 17.5 tokens.

Accuracy results for the NEGRA corpus are the lowest. Therefore, we expect the effect on annotation times to be even larger for the other corpora.

7.5.2 Learning Curves

The accuracy of a parser heavily depends on the amount of training data available. Usually, the larger the training corpus, the better the parameter estimates and the better the parsing results.

This section presents the learning curves of Cascaded Markov Models in the interactive annotation mode. We show the accuracy depending on the amount of training data. The curves for the four corpora are shown in figures 7.26 to 7.29. Training length is the number of tokens at the word level used for training. Each training length was tested ten times, training and test sets were disjoint, results were averaged. The training length is given on a logarithmic scale.

The diagrams show the percentage of cases in which a new phrase was suggested, the percentage of correct suggestions and the percentage of cases in which either the first or second suggestion was correct.

For the NEGRA corpus, we expect higher accuracies when further increasing

annotators where not trained.

Table 7.18: Increase in accuracy for the interactive annotation task when doubling the training size from half of the corpus size to the full size (minus test set). This is used to roughly estimate the gain in accuracy when further increasing the training size.

Corpus	from	to	Δaccuracy
NEGRA corpus	130,000	260,000	1.84
Penn Treebank	500,000	1,000,000	0.06
Verbmobil (German)	50,000	100,000	0.26
Verbmobil (English)	70,000	140,000	0.02

the size of the training set, while for the other corpora, the curves are already very flat at the given maximum sizes. Table 7.18 shows the increase in accuracy when doubling the training size from half of the corpus size to the full size.

For the Verbmobil Corpora, accuracy is very high even for very small training sets with just 100 or 200 tokens.

7.5.3 Summary of Interactive Results

For the interactive annotation task, we achieve accuracies of 70% for the NEGRA corpus, 77% for the Penn Treebank, 92% for the German part of the Verbmobil Corpus, and 83% for the English part. Taking the second best alternative into account, accuracies reach 84 – 97%.

The accuracies are sufficient to facilitate corpus annotation and to speed up the annotation process. For the NEGRA corpus, which yields worst results for this task, we measured a 30% speed-up when comparing human annotation times with and without using automatic generation of structures by Cascaded Markov Models.

Except for the NEGRA corpus, accuracies for the interactive annotation task are high even with small amounts of training data. With just 1,000 tokens of training, we achieve 65% for the Penn Treebank, 79% for Verbmobil (English) and 88% for Verbmobil (German).

Accuracies for the NEGRA corpus are lower than for the other corpora. This as probably due to the very flat annotation scheme and the absence of unary productions, which are used in the other three corpora.

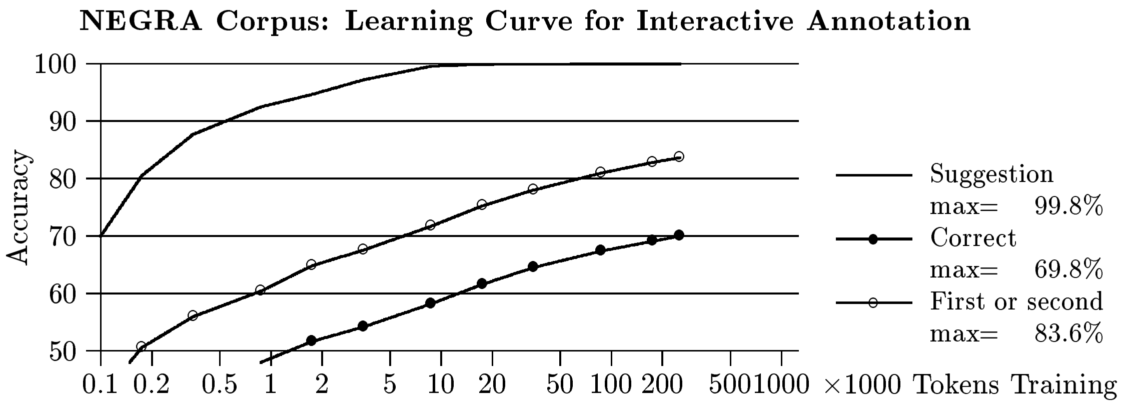


Figure 7.26: Learning curves for the interactive annotation task using the NEGRA corpus (see below for a description).

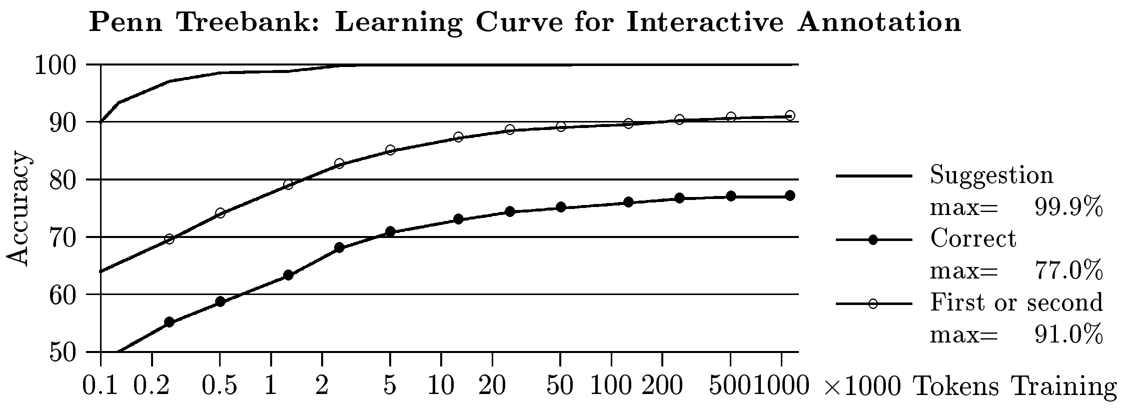


Figure 7.27: Learning curves for the interactive annotation task using the Penn Treebank. The diagram shows, depending on the size of the training set, the percentage of cases in which a suggestion is made, the percentage of correct suggestions, and the percentage of cases in which either the first or the second suggestion is correct. The diagram shows unlabeled accuracies. The training sets of variable sizes as well as test sets of around 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

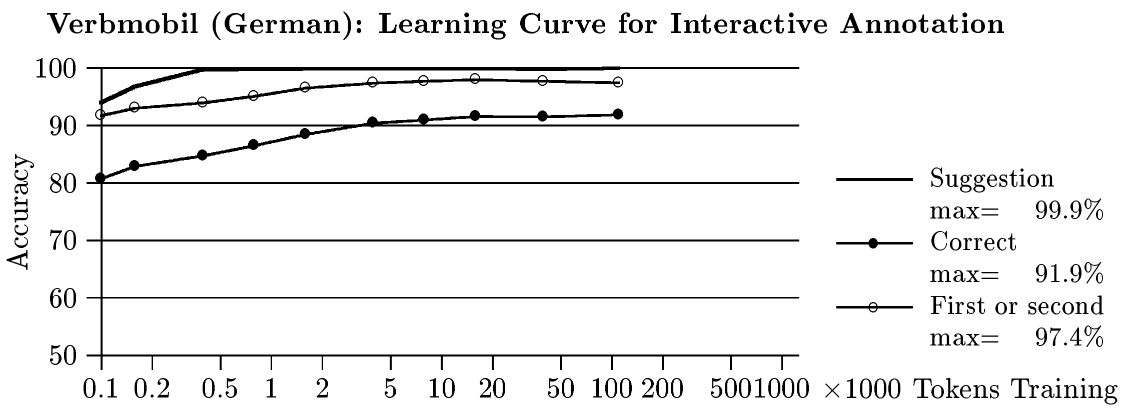


Figure 7.28: Learning curves for the interactive annotation task using the German part of the Verbmobil corpus (see below for a description).

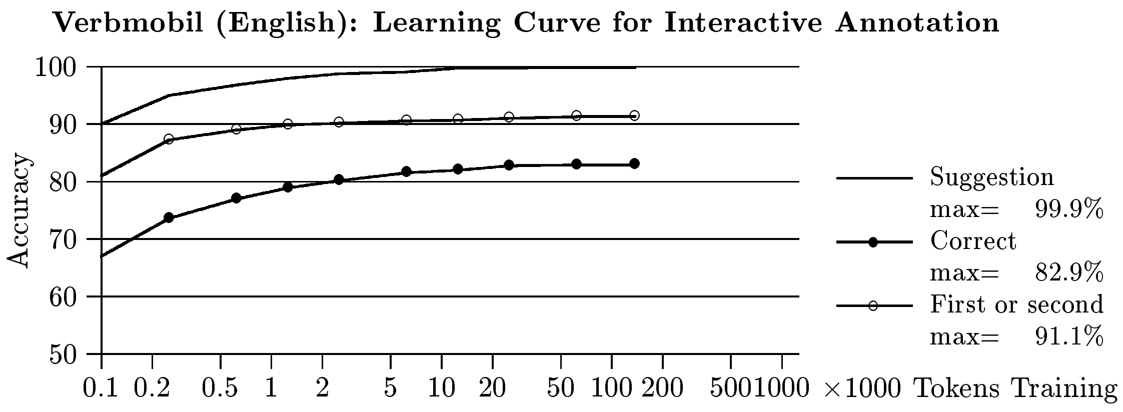


Figure 7.29: Learning curves for the interactive annotation task using the English part of the Verbmobil corpus. The diagram shows, depending on the size of the training set, the percentage of cases in which a suggestion is made, the percentage of correct suggestions, and the percentage of cases in which either the first or the second suggestion is correct. The diagram shows unlabeled accuracies. The training sets of variable sizes as well as test sets of around 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

7.6 Partial Parsing with Cascaded Markov Models

This section presents partial parsing (chunking) results for applying Cascaded Markov Models, using the uncorrected output of a layer as the input for the next layer.

Chunks are extracted from the four corpora by removing all S nodes, VP nodes and coordinations, and by detaching postnominal PPs and appositions from NPs and PPs. For the German corpora, we additionally detached prenominal adverbial modifiers from NPs and PPs. The resulting corpora contain base NPs and PPs together with the structure of their constituents (figure 1.2 on page 11 shows an example extracted from the NEGRA corpus).

From table 7.19, we see that the number of Markov Models that are necessary for processing the chunked corpora is rather small. The avarage depth of chunks ranges from 1.5 to 3.1 layers, and we need 3 to 7 layers to cover 99% of all sentences.

We will first determine partial parsing recall and precision depending on the number of Markov Models that are employed and then present results depending on the size of the training set.

7.6.1 Partial Parsing Results

Figures 7.30 to 7.33 show recall and precision when using different numbers of Markov Models for partial parsing. The number of Markov Models limits the maximum depth of structures that can be recognized. The experiments use the (possibly ambiguous) output of a part-of-speech tagger at layer 0. Part-of-speech accuracy is indicated at the bottom line of each diagram.

We see that recall increases with the number of Markov Models, but at the same time, precision decreases. This indicates that nodes at lower layers are easier to parse, which can be due to two reasons. First, errors at different layers are not independent, so wrong structures at lower layers may result in wrong structures at higher layers. Second, nodes at higher layers are usually more complex and therefore harder to recognize.

Precision for the first layer is between 91 and 98%, and it drops to values between 88 and 94% when handling more layers. Recall for small numbers of layers is very

Table 7.19: Depths of chunks. The table shows the number of layers of annotations in the four corpora (excluding the part-of-speech layer) that are used for chunks (i.e., mainly NP/PP nodes and their consituents). It shows the average number, the number of layers that is necessary to cover 99% of the sentences, and the maximum depth found in the corpus.

	number of layers		
corpus	avg.	99%	max.
NEGRA corpus	1.5	5	11
Penn Treebank	3.1	4	8
VM German	2.0	3	10
VM English	2.8	7	13

low, since all nodes with depths larger than the specified number of layers cannot be recognized. For larger numbers of layers recall reaches around 85 – 91%.

7.6.2 Learning Curves

Recall and precision of a parser heavily depend on the amount of training data available. Usually, the larger the training corpus, the better the parameter estimates and the better the parsing results.

This section presents the learning curves when using Cascaded Markov Models for partial parsing. We show recall and precision depending on the amount of training data. The curves for the four corpora are shown in figures 7.34 to 7.37. Training length is the number of tokens at the word level used for training. Each training length was tested ten times, training and test sets were disjoint, results were averaged. The training length is given on a logarithmic scale.

For the Penn Treebank and the English part of the Verbmobil corpus, the curves are very flat when reaching the maximum training size (i.e., 90% of the corpus size). For the NEGRA corpus and the German part of the Verbmobil corpus, we expect significantly higher accuracies when further increasing the size of the training set. The diagram for the German Verbmobil data is the only one that starts with a precision lower than the recall, for all others precision is always higher than recall. We do not have an explanation for this effect.

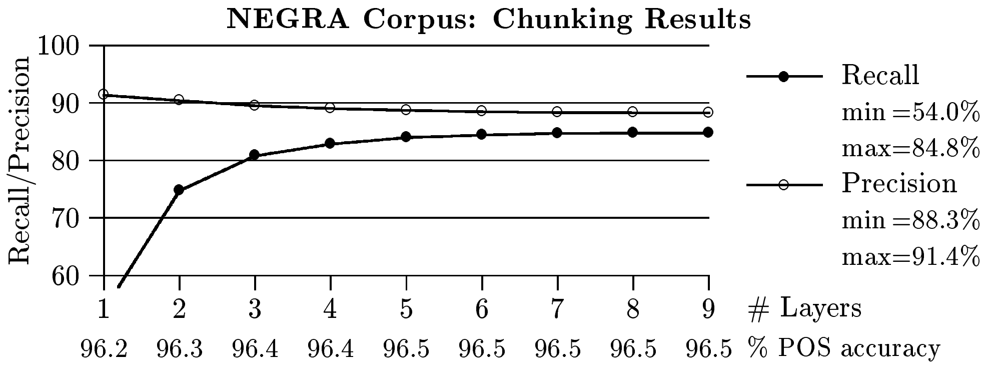


Figure 7.30: Chunking results for the NEGRA Corpus (see below for a description)

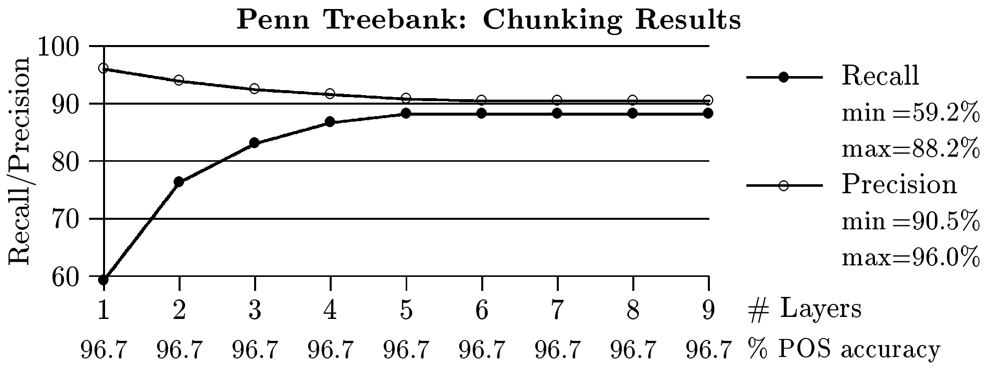


Figure 7.31: Chunking results for the Penn Treebank. The diagram shows recall and precision depending on the number of layers that are used for parsing. Layer 0 is used for part-of-speech tagging, for which tagging accuracies are given at the bottom line. The corpus was divided into 90% for training and 10% for testing, the results were averaged over 10 test runs.

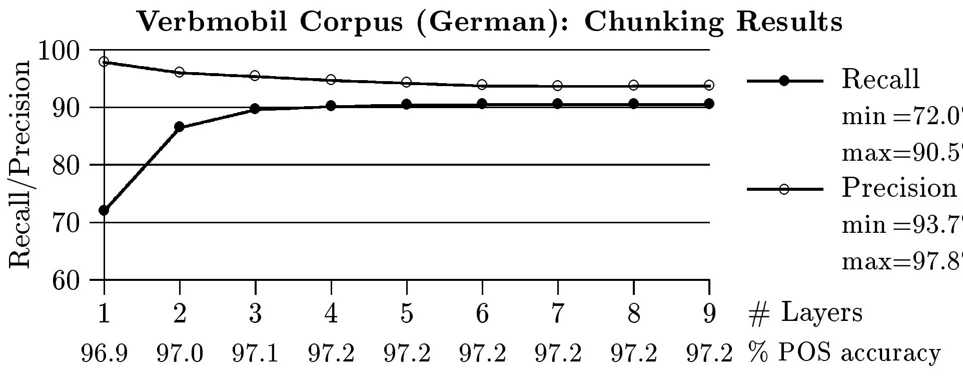


Figure 7.32: Chunking results for the German part of the Verbmobil Corpus (see below for a description).

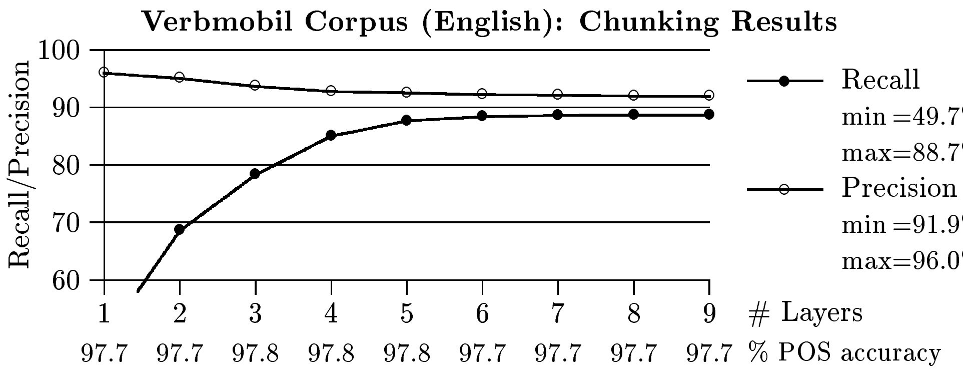


Figure 7.33: Chunking results for the German part of the Verbmobil Corpus. The diagram shows recall and precision depending on the number of layers that are used for parsing. Layer 0 is used for part-of-speech tagging, for which tagging accuracies are given at the bottom line. The corpus was divided into 90% for training and 10% for testing, the results were averaged over 10 test runs.

Table 7.20: Increase in precision for the interactive annotation task when doubling the training size from half of the corpus size to the full size (minus test set). This is used to roughly estimate the gain in accuracy when further increasing the training size.

Corpus	from	to	Δprecision
NEGRA corpus	130,000	260,000	1.07
Penn Treebank	500,000	1,000,000	0.27
Verbmobil (German)	50,000	100,000	1.29
Verbmobil (English)	70,000	140,000	0.42

Table 7.20 shows the gain in precision when doubling the size of the corpora. We expect significant effects when further increasing the size of the NEGRA corpus and the German Verbmobil corpus. Increase is smaller for the Penn Treebank and the English Vermobil corpus.

7.6.3 Summary of Partial Parsing Results

Chunking results range from 85 to 91% for recall and from 88 to 94% for precision, using small numbers (3 – 7) of layers. This includes a layer 0 for part-of-speech tagging, which yields accuracies in the range 96.2 – 97.7%. The exact results depend on the corpus type, the language, and the annotation scheme.

Chunking results for Penn Treebank data were reported earlier by (Ramshaw & Marcus, 1995), (Argamon et al., 1998), (Cardie & Pierce, 1998), and (Veenstra, 1998). They reported recall in the range 91.1 – 94.3% and precision in the range 90.7 – 91.8%. Our results for the Penn Treebank are slightly below these values. This is most probably due to the harder task in our experiments. We additionally recognize PPs, which includes the disambiguation of prepositions, and we additionally recognize sub-structures of constituents instead of marking NP boundaries.

Chunking results for the NEGRA corpus using structural tags are presented by Skut (forthcoming). He reports approx. 87% recall and 89% precision for a maximum entropy model. This is slightly higher than our result, but his model processed correctly pre-tagged text while our model takes untagged text as input.

For the Penn Treebank and the Verbmobil corpora, only a few thousand tokens

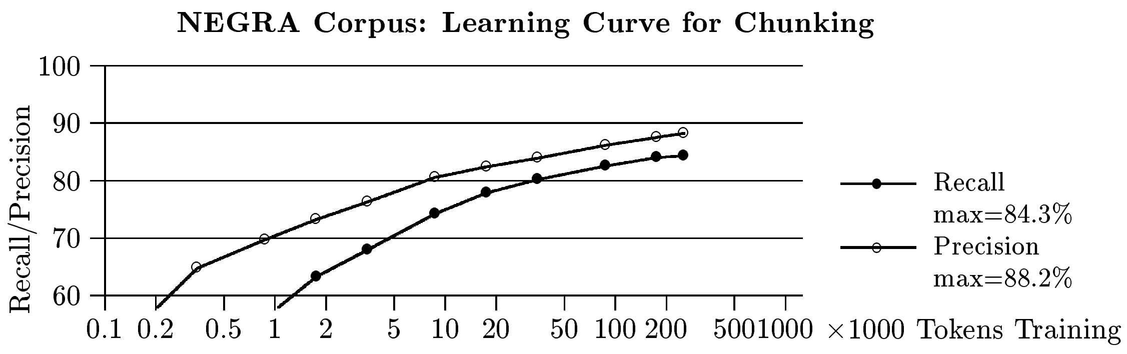


Figure 7.34: Learning curves for chunking the NEGRA corpus using 5 layers (see below for a description).

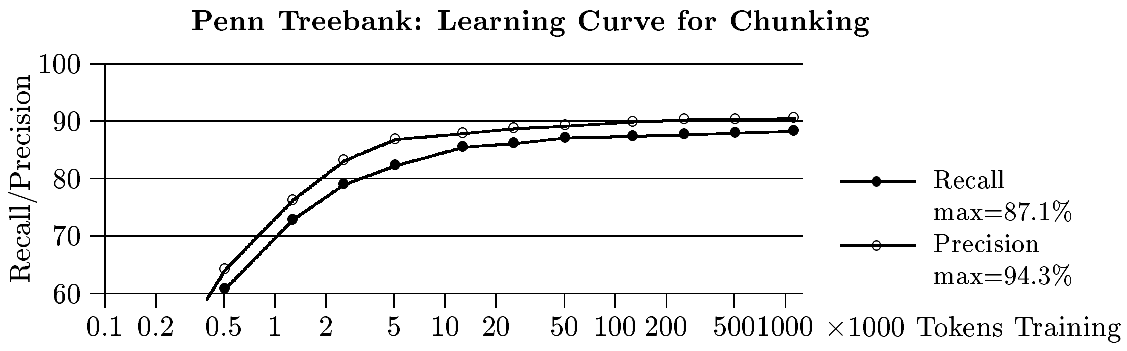


Figure 7.35: Learning curves for chunking the Penn Treebank. The diagram shows recall and precision depending on the amount of training data when using 9 layers of Markov Models plus one layer for part-of-speech tagging. The training sets of variable sizes as well as test sets of around 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

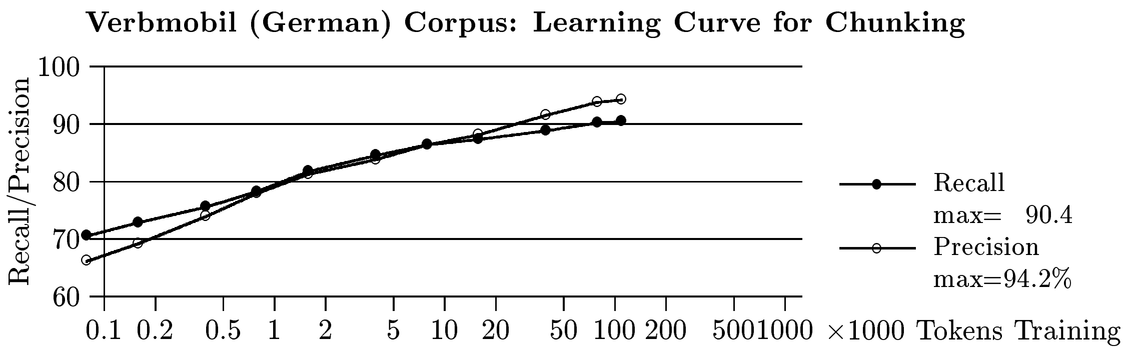


Figure 7.36: Learning curves for chunking the German part of the Verbmobil Corpus (see below for a description).

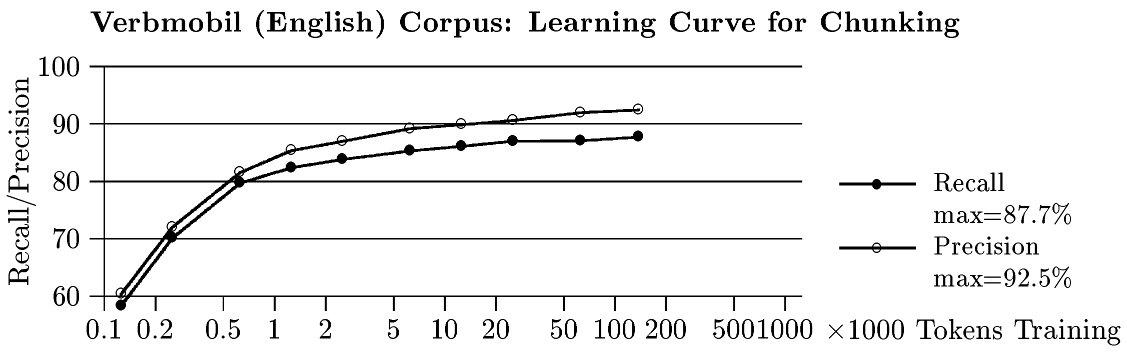


Figure 7.37: Learning curves for chunking the German part of the Verbmobil Corpus. The diagram shows recall and precision depending on the amount of training data when using 5 layers of Markov Models plus one layer for part-of-speech tagging. The training sets of variable sizes as well as test sets of around 10,000 tokens were randomly chosen. Training and test sets were disjoint, the procedure was repeated 10 times and results were averaged.

of training are sufficient to achive recall and precision rates between 80 and 90%. For the NEGRA corpus, more training data is necessary which is most probably due to the very flat annotation yielding a larger number of context-free rules.

Chapter 8

Conclusions

8.1 Contributions

In this thesis, we have presented new methods for robust syntactical language processing. The starting point was statistical part-of-speech tagging. This technique was improved and extended to new types of syntactical processing: the assignment of grammatical functions and the assignment of phrase categories to existing structures.

Furthermore, we have exploited the probability value of assignments, so that it is not only possible to leave ambiguity in the output for selected elements, but also to quantify the expected accuracy depending on the probability. In this case, the output for each word or phrase is a probability distribution over tags or labels, which is more informative than just the best assignment and enables better communication with following processing steps.

We introduced structural processing with Cascaded Markov Models. The model is inspired by finite state cascades. It has the additional advantages that probabilities are taken into account and that the corresponding grammar is learned from a corpus (instead of using hand-crafted rules). With Cascaded Markov Models, each layer of a syntactic structure is represented by a Markov Model, and a lower layer passes its possibly ambiguous output together with a probability distribution as input to the next higher layer.

Cascaded Markov Models introduce a new syntactic processing model and exploit a type of information that is absent in previously presented parsers for context-free structures: left-to-right transitions of non-terminal nodes, regardless of their position in the hierarchical structure.

A new interactive treebank annotation mode was presented, using part-of-speech tagging, tagging grammatical functions, assigning phrase categories, and Cascaded Markov Models. This interactive mode enables very efficient syntactic annotation.

All techniques have been intensively evaluated on written and transliterated spoken corpora in English and German and yield good results. Furthermore, their practical usefulness has been demonstrated in several projects that have created (and are still extending) corpus resources.

We presented learning curves, i.e., diagrams of amount of training data vs. accuracy, for the different tasks and corpora. These curves indicate that all presented methods can be already successfully used with very small amounts of training data (1,000 words or even less). This is an important feature for methods used in a corpus annotation process, starting with virtually no annotated data and extending the corpus incrementally. The learning curves become very flat when using the full corpus sizes for training (100,000 – 1 million words), and we expect only very small increases in accuracy when further increasing the size of the training corpora. Exceptions are the annotation task for the NEGRA corpus, and the chunking tasks for the NEGRA and German Verbmobil corpora. For these, significant improvements can be expected when, e.g., doubling the size of the training corpus.

8.2 Future Directions

The presented methods improve and extend existing ones, but nevertheless there is a lot of room for further developments. First of all, the investigation of the new parsing model concentrates on the extension of Markov Models and the better use of local context information when generating syntactic structures. This can be complemented by lexical information. Several investigations have shown that lexicalized grammars yield better results than non-lexicalized grammars. In our model, each state of a Cascaded Markov Model emits a partial parse tree, including all lexical information. But due to sparse data, we restricted probability estimates to consider only very rudimentary information about the terminal string (i.e., we pass only rudimentary morphological information to higher nodes). We expect that complementing *contextualization* with *lexicalization*, which are to some extent orthogonal, will further improve results. Yet, methods for their combination need to be developed. More generally, the way of encoding syntactic information is crucial and influences performance. This concerns the encoding of informations in the nodes of syntactic structures as well as the structural encoding itself. For instance, it is currently unclear whether deep or flat structures are better suited for automatic processing.

Another line of investigation is concerned with parameter estimation. We presented the generation of parameters from annotated corpora and used linear interpolation for smoothing. While we do not expect improvements by re-estimation on raw data, other smoothing methods may result in better accuracies. Several investigations have shown that parameter estimation within the maximum entropy framework yields better results than linear interpolation for small training sizes. Yet, the high complexity of maximum entropy parameter estimation requires research on the method itself as well as investigations on feature selection, so that relevant linguistic features can be manually pre-selected.

An interesting open question is the possibility of combining several layers of a Cascaded Markov Model into one large model. Since the generated structure is part of the output of the complete process, it is important that the results of all layers are preserved in the output of the combined model. Additionally, it has to be taken into account that the output of a layer corresponds to the states of a Markov Model. Therefore, current findings on weighted finite state transducers are not directly applicable but may play an important role in the solution of this problem.

Several methods have been proposed for maintaining an agenda for statistical chart parsing. A *figure of merit* indicates the expected value of an edge in the chart, and the best edge is added first. We expect that such a parser can benefit from a figure of merit calculated by Cascaded Markov Models.

Another line of investigation is concerned with the parsing of discontinuous constituents. Cascaded Markov Models as presented here exploit transition probabilities of adjacent words and phrases. It is a straight-forward extension to allow the models to skip one or more elements and continue with a later word or phrase, thereby forming discontinuous constituents. At least for languages with relatively free word order, grammars allowing discontinuous constituents are better suited than grammars without this feature. Therefore, the NEGRA corpus as well as the Prague Dependency Treebank are annotated with discontinuous constituents. Building a parser for these treebanks is a challenging task.

Not only the parsing model itself is an interesting topic for further investigations, but also the relation to psycholinguistics. Early investigations already suggested the use of Markov Models in psycholinguistic theories (Osgood, 1963). Currently, an increasing number of investigations is concerned with frequency-based psycholinguistic models. With a small change to the presented methods, Cascaded Markov Models can process sentences incrementally. Instead of building layer by layer sequentially, all layers are built in parallel as soon as a new input word arrives. Distance or recency is not explicitly modeled, but it plays an important role because Markov Models only take into account a finite history. This type of processing exhibits similarities with the processing model presented by Kempen and Vosse (Kempen & Vosse, 1987; Vosse & Kempen, 1991; Kempen, 1996) and at the same time fits into a modular statistical architecture as investigated by Crocker and Corley (to appear). There is clear evidence for a connection between human language processing and stochastic processes, but more sophisticated processing methods are needed to establish the connection between parsing and psycholinguistics.

References

Abney, S. (1990). Rapid incremental parsing with repair. In *Proceedings of the 6th new OED conference: Electronic text research.* Waterloo, Ontario.

Abney, S. (1991). Parsing by chunks. In R. Berwick, S. Abney, & C. Tenny (Eds.), *Principle-based parsing.* Dordrecht: Kluwer Academic Publishers.

Abney, S. (1996). Partial parsing via finite-state cascades. In *Proceedings of the ESSLLI workshop on robust parsing.* Prague, Czech Republic.

Aho, A. V., & Ullman, J. D. (1972). *The theory of parsing, translation and computing.* New Jersey: Prentice-Hall.

Appelt, D., Hobbs, J., Bear, J., Israel, D. J., & Tyson, M. (1993). FASTUS: a finite-state processor for information extraction from real-world text. In *Proceedings of IJCAI-93.* Washington, DC.

Argamon, S., Dagan, I., & Krymolowski, Y. (1998). A memory-based approach to learning shallow natural language patterns. In *Proceedings of the 17th international conference on Computational Linguistics COLING-ACL-98).* Montreal, Canada.

Armstrong, S., Russell, G., Petitpierre, D., & Robert, G. (1995). An open architecture for multilingual text processing. In *Proceedings of the ACL SIGDAT workshop. From texts to tags: Issues in multilingual language analysis.* Dublin, Ireland.

Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occuring in the statistical analysis of probabilistic functions in Markov chains. *The Annals of Mathematical Statistics*, *41*, 164-171.

Becker, T. (1994). *Hytag: A new type of tree adjoining grammars for hybrid syntactic representation of free word order languages.* Unpublished doctoral dissertation, Saarbrücken.

Bemová et al. (1997). *Anotace na analytické rovine (analytical level annotation)* (Technical Report No. #3). Prague: IFAL MFF UK.

Berger, A. L., Della Pietra, S. A., & Della Pietra, V. J. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, *22*(1), 39–71.

Black, E., Jelinek, F., Lafferty, J., Magerman, D., Mercer, R., & Roukos, S. (1993). Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 31st conference of the Association of Computational Linguistics ACL-93.*

Black, Ezra, et al. (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the DARPA speech and natural language workshop* (p. 306-311). Columbus, Ohio.

Bod, R. (1993). Data-oriented parsing as a general framework for stochastic language processing. In *Proceedings of twente workshop on language technology: Parsing natural language.* Twente.

Bod, R. (1995). *Enriching linguistics with statistics: Performance models of natural language.* Universiteit van Amsterdam: Institute for Logic, Language and Computation.

Bod, R., & Scha, R. (1994). Prediction and disambiguation by means of data-oriented parsing. In *Proceedings of twente workshop on language technology.* Twente.

Brants, T. (1995). Tagset reduction without information loss. In *Proceedings of the 33rd annual meeting of the Association for Computational Linguistics.* Cambridge, MA.

Brants, T. (1996a). Better language models with model merging. In *Proc. of the conference on empirical methods in natural language processing.* Philadelphia, PA.

Brants, T. (1996b). *TnT – a statistical part-of-speech tagger* (Technical Report). Universität des Saarlandes, Computational Linguistics.

Brants, T. (1999a). Cascaded markov models. In *Proceedings of 9th conference of the European chapter of the Association for Computational Linguistics EACL-99.* Bergen, Norway.

Brants, T. (1999b). *TnTL – statistical tagging of lattices* (Technical Report). Universität des Saarlandes, Computational Linguistics.

Brants, T., & Samuelsson, C. (1995). Tagging the teleman corpus. In *Proceedings of the 10th nordic conference of computational linguistics NODALIDA-95.* Helsinki, Finland. (Also available as CLAUS Report 54)

Brants, T., & Skut, W. (1998). Automation of treebank annotation. In *Proceedings of new methods in language processing NeMLaP-98.* Sydney, Australia.

Brill, E. (1993). *A corpus-based approach to language learning.* University of Pennsylvania: Ph.D. Dissertation, Department of Computer and Information Science.

Brown, P. F., Pietra, V. J. D., deSouza, P. V., Lai, J. C., & Mercer, R. L. (1992). Class-based n-gram models of natural language. *Computational Linguistics, 18(4)*, 467-479.

Cardie, C., & Pierce, D. (1998). Error-driven pruning of treebank grammars for base noun phrase identification. In *Proceedings of the 17th international conference on Computational Linguistics COLING-ACL-98).* Montreal, Canada.

Carter, D. (1997). The TreeBanker: a tool for supervised training of parsed corpora. In *ACL workshop: Computational environments for grammar development and linguistic engineering.* Madrid, Spain.

Charniak, E. (1993). *Statistical language learning.* Cambridge, MA: MIT Press.

Church, K. W. (1988). A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the second conference on applied natural language processing ANLP-88* (p. 136-143). Austin, Texas, USA.

Cole, R. A., Mariani, J., Uszkoreit, H., Zaenen, A., & Zue, V. (Eds.). (1998). *Survey of the state of the art in human language technology.* Cambridge University Press.

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of ACL-96.* Santa Cruz, CA, USA.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL/EACL-97.* Madrid, Spain.

Crocker, M. W., & Corley, S. (to appear). Modular architectures and statistical mechanisms: The case from lexical category disambiguation. In Merlo & Stevenson (Eds.), *The lexical basis of sentence processing.* John Benjamins.

Cutting, D., Kupiec, J., Pedersen, J., & Sibun, P. (1992). A practical part-of-speech tagger. In *Proceedings of the 3rd conference on applied natural language processing (ACL)* (p. 133-140).

Daelemans, W., Zavrel, J., Berck, P., & Gillis, S. (1996). Mbt: A memory-based part of speech tagger-generator. In *Proceedings of the workshop on very large corpora.* Copenhagen, Denmark.

Declerck, T., Klein, J., & Neumann, G. (1998). Evaluation of the nlp components of an information extraction system for german. In *Proceedings of the conference on language resources and evaluation (lrec).* Granada, Spain.

DeRose, S. J. (1988). Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, *14*(1), 31-39.

Eisner, J. M. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the international conference on computational linguistics COLING-96.* Copenhagen, Denmark.

Elworthy, D. (1994). Does Baum-Welch re-estimation help taggers? In *Proceedings of the fourth conference on applied natural language processing ANLP-94.* Stuttgart, Germany.

Elworthy, D. (1995). Tagset design and inflected languages. In *Proceedings of the workshop on very large corpora* (p. 1-9). Dublin, Ireland.

Finkler, W., & Neumann, G. (1988). *MORPHIX – a fast realization of a classification-based approach to morphology* (Bericht Nr. 40). Universitä des Saarlandes, SFB 314.

Francis, N. W., & Kucera, H. (1982). *Frequency analysis of English usage.* Boston: Houghton Mifflin.

Gale, W. A., & Church, K. W. (1990). Poor estimates of context are worse than none. In *Proceedings of the speech and natural language workshop* (p. 283-287). Hidden Valley, PA.

Godfrey, McDaniel, & Holliman. (1993). SWITCHBOARD: A telephone speech corpus for research and development. In *Proceedings of the IEEE international conference on acoustics, speech and signal processing ICASSP-93.* Minneapolis, MN.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, *40*, 237-264.

Goodman, J. (1996). Efficient algorithms for parsing the dop model. In *Proceedings of the conference on empirical methods in natural language processing* (p. 143-152). Philadelphia, PA.

Goodman, J. (1998). *Parsing inside-out.* Unpublished doctoral dissertation.

Greenbaum, S. (Ed.). (1996). *Comparing English worldwide: The international corpus of English.* Oxford: Clarendon Press.

Grinberg, D., Lafferty, J., & Sleator, D. (1995). A robust parsing algorithm for link grammars. In *Proceedings of the fourth international workshop on parsing technologies.* Prague, Czech Republic.

Gross, M. (1997). The construction of local grammars. In E. Roche & Y. Schabes (Eds.), *Finite-state language processing.* Cambridge, MA: MIT-Press.

Haapalainen, M., & Majorin, A. (1995). GERTWOL und Morphologische Disambiguierung für das Deutsche. In *Proceedings of the 10th nordic conference of computational linguistics NODALIDA-95.*

Hajic, J. (1998). Building a syntactically annotated corpus: The prague dependency treebank. In *Issues of valency and meaning.* Praha: Karolinum.

Hajic, J., & Hladka, B. (1998). Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of the international conference on computational linguistics COLING-98* (p. 483-490). Montreal, Canada.

Halteren, H. van, Zavrel, J., & Daelemans, W. (1998). Improving data driven wordclass tagging by system combination. In *Proceedings of the international conference on computational linguistics COLING-98* (p. 491-497). Montreal, Canada.

Harris, Z. (1962). *String analysis of language structure.* The Hague, Netherlands: Mouton and Co.

Hindle, D. (1983). Deterministic parsing of syntactic non-fluencies. In *Proceedings of the 21st annual meeting of the association for computational linguistics ACL-83.* Cambridge, MA.

Jelinek, F., Lafferty, J., Magerman, D., Mercer, R., Ratnaparkhi, A., & Roukos, S. (1994). Decision tree parsing using a hidden derivation model. In *Proceedings of the ARPA workshop on human language technology.* San Francisco, CA: Morgan Kaufmann.

Jelinek, F., & Mercer, R. L. (1980). Interpolated estimation of Markov source parameters from sparse data. In *Pattern recognition in practice* (p. 381-397). North Holland.

Johnson, M. (1998). The effect of alternative tree representations on tree bank grammars. In *Proceedings of new methods in language processing NeMLaP-98.* Sydney, Australia.

Joshi, A. K. (1960). *Advances in documentation and library science* (Vols. III, part 2). New York: Interscience Publishers, Inc.

Joshi, A. K., & Hopely, P. (1997). A parser from antiquity. *Natural Language Engineering, 2(4).*

Joshi, A. K., & Srinivas, B. (1994). Disambiguation of super parts of speech (or supertags). In *Proceedings of the international conference on computational linguistics COLING-94.* Kyoto, Japan.

Karlsson, F., Voutilainen, A., Heikkilä, J., & Anttila, A. (1994). *Constraint grammar: A language-independent system for parsing unrestricted text.* Berlin: Mouton de Gruyter.

Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing, 35*(3).

Kempe, A. (1997). Finite state transducers approximating hidden markov models. In *Proceedings of the 35th annual meeting of the association for computation linguistics ACL-97* (p. 460-467). Madrid, Spain.

Kempen, G. (1996). Computational models of syntactic processing in human language comprehension. In A. Dijkstra & K. D. Smedt (Eds.), *Computational psycholinguistics: symbolic and subsymbolic models of language processing* (p. 192-220). London: Taylor and Francis.

Kempen, G., & Vosse, T. (1987). Incremental syntactic tree formation in human sentence processing: A cognitive architecture based on activation decay and simulated annealing. *Connection Science, 1*, 273-290.

Krenn, B., & Samuelsson, C. (1997). *The linguist's guide to statistics – don't panic* (Course Compendium). Saarbrücken: University of the Saarland and Bell Laboratories.

Kupiec, J. M. (1992). Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language, 6*, 225-242.

Lafferty, J., Sleator, D., & Temperley, D. (1992). Grammatical trigrams: A probabilistic model of link grammar. In *Proceedings of the AAAI conference on probabilistic approaches to natural language.* Cambridge, MA.

Lezius, W. (1996). Morphologiesystem MORPHY. In R. Hausser (Ed.), *Linguistische Verifikation: Dokumentation zur ersten Morpholympics 1994.* Tübingen: Niemeyer.

Lima, E. de. (1997). Assigning grammatical relations with a back-off model. In *Proceedings of the conference on empirical methods in natural language processing EMNLP-97.* Providence, RI, USA.

Magerman, D. (1994). *Natural language parsing as statistical pattern recognition.* Unpublished doctoral dissertation, Stanford University.

Magerman, D. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting of the Association for Computational Linguistics ACL-95.* Cambridge, MA.

Magerman, D. M., & Weir, C. (1992). Probabilistic prediction and picky chart parsing. In *DARPA speech and natural language workshop* (p. 128-133). Arden House, NY.

Marcus, M., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, *19*(2), 313-330.

Markov, A. A. (1913). Russian [an example of statistical investigation in the text of 'Eugene Onyegin' illustrating coupling of 'tests' in chains]. *Proceedings of the Academy of Sciences, St. Petersburg*, *7, VI*, 153-162.

Marshall, I. (1983). Choice of grammatical word-class without global syntactic analysis. *Computers in the Humanities*, *17*, 139-150.

Merialdo, B. (1993). Tagging english text with a probabilistic model. *Computational Linguistics*, *20(2)*, 155-172.

Osgood, C. E. (1963). On understanding and creating sentences. *American Psychologist*, *18*(12), 735-751.

Pechmann, T., Uszkoreit, H., Engelkamp, J., & Zerbst, D. (1994). *Word order in the german middle field* (CLAUS-Report / Nr. 43, August 1994 No. 43). Saarbrücken: Universität des Saarlandes-CL.

Pereira, F., Riley, M., & Sproat, R. (1994). Weighted rational transductions and their application to human language processing. In *Proceedings of the workshop on human language technology.* San Francisco, CA: Morgan Kaufmann.

Pereira, F., & Schabes, Y. (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th annual meeting of the association for computational linguistics ACL-92.* Newark, Delaware.

Pereira, F. C. N., & Riley, M. D. (1997). Speech recognition by composition of weighted finite automata. In E. Roche & Y. Schabes (Eds.), *Finite-state language processing.* Cambridge, MA: MIT-Press.

Plaehn, O. (1998). *Das NEGRA-Annotations-Tool: Users manual* (Technical Report). Saarbrücken, Germany: Saarland University, Computational Linguistics.

Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE* (Vol. 77, pp. 257–285).

Ramshaw, L. A., & Marcus, M. P. (1995). Text chunking using transformation-based learning. In *Proceedings of the third workshop on very large corpora.* Dublin, Ireland.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing EMNLP-96.* Philadelphia, PA.

Ratnaparkhi, A. (1997). A linear observed time statistical parser based on maximum entropy models. In *Proceedings of the conference on empirical methods in natural language processing EMNLP-97.* Providence, RI.

Roche, E. (1992). Text disambiguation by finite state automata, an algorithm and experiments on corpora. In *Proceedings of the 14th international conference on computational linguistics COLING-92* (p. 993-997). Nantes, France.

Roche, E. (1994). Two parsing algorithms by means of finite state transducers. In *Proceedings of the 15th international conference on computational linguistics COLING-94* (p. 431-435). Kyoto, Japan.

Roche, E. (1997). Parsing with finite-state transducers. In E. Roche & Y. Schabes (Eds.), *Finite-state language processing.* Cambridge, MA: MIT-Press.

Roche, E., & Schabes, Y. (1995). Deterministic part-of-speech tagging with finite-state transducers. *Computational Linguistics*, *21*, 227-253.

Roche, E., & Schabes, Y. (Eds.). (1997). *Finite-state language processing.* Cambridge, MA: MIT-Press.

Sampson, G. (1995). *English for the computer.* Oxford: Oxford University Press.

Samuelsson, C. (1993). Morphological tagging based entirely on Bayesian inference. In *9th nordic conference on computational linguistics NODALIDA-93.* Stockholm University, Stockholm, Sweden.

Samuelsson, C. (1996). Handling sparse data by successive abstraction. In *Proceedings of the 16th international conference on computational linguistics COLING-96.* Copenhagen, Denmark.

Samuelsson, C. (1997). Extending n-gram tagging to word graphs. In *Proceedings of the 2nd international conference on recent advances in natural language processing RANLP-97.* Tzigov Chark, Bulgaria.

Schabes, Y., Abeillé, A., & Joshi, A. K. (1988). Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *The 12th international conference on computational linguistics COLING-88.* Budapest, Hungary.

Schabes, Y., & Waters, R. (1993). Stochastic lexicalized context-free grammars. In *Proceedings of the third international workshop on parsing technologies.* Tilburg/Durbuy.

Schmid, H. (1995). Improvements in part-of-speech tagging with an application to German. In H. Feldweg & E. Hinrichts (Eds.), *Lexikon und Text.* Niemeyer, Tübingen.

Schneider, G., & Volk, M. (1998). Adding manual constraints and lexical look-up to a brill-tagger for german. In *Proceedings of the ESSLLI workshop on recent advances in corpus annotation.* Saarbrücken, Germany.

Schütze, H. (1995). Distributional part-of-speech tagging. In *Proceedings of 7th conference of the european chapter of the association for computational linguistics EACL-95.* Dublin, Ireland.

Sekine, S. (1997). The domain dependence of parsing. In *Proceedings of the fifths conference on applied natural language processing ANLP-97* (p. 96-102). Washington, D.C.

Sekine, S. (1998). *Corpus-based parsing and sublanguage studies.* Unpublished doctoral dissertation, New York University.

Sekine, S., & Grishman, R. (1995). A corpus-based probabilistic grammar with only two non-terminals. In *Proceedings of the fourth international workshop on parsing technologies* (p. 216-223). Prague, Czech Republic.

Silberztein, M. D. (1993). *Dictionnaires electroniques et analyse lexicale du francais – le système INTEX.* Masson.

Silberztein, M. D. (1997). The lexical analysis of natural language. In E. Roche & Y. Schabes (Eds.), *Finite-state language processing.* Cambridge, MA: MIT-Press.

Sima'an, K. (1996a). Computational complexity of probabilisitc disambiguation by means of tree-grammars. In *Proceedings of the international conference on computational linguistics COLING-96.* Copenhagen, Denmark.

Sima'an, K. (1996b). An optimized algorithm for data oriented parsing. In *Recent advances in natural language processing 1995. Volume 136 of current issues in linguistic theory.* Amsterdam, The Netherlands: John Benjamins.

Skut, W. (forthcoming). *Partial parsing for corpus annotation and text processing.* Unpublished doctoral dissertation, Saarland University.

Skut, W., & Brants, T. (1998). Chunk tagger – statistical recognition of noun phrases. In *Proceedings of the ESSLLI workshop on automated acquisition of syntax and parsing.* Saarbrücken, Germany.

Skut, W., Brants, T., Krenn, B., & Uszkoreit, H. (1997). Annotating unrestricted german text. In *Fachtagung der Sektion Computerlinguistik der Deutschen Gesellschaft für Sprachwissenschaft.* Heidelberg, Germany.

Skut, W., Krenn, B., Brants, T., & Uszkoreit, H. (1997). An annotation scheme for free word order languages. In *Proceedings of the fifth conference on applied natural language processing ANLP-97.* Washington, DC.

Sleator, D., & Temperley, D. (1993). Parsing English with a link grammar. In *Proceedings of the third international workshop on parsing technologies.* Tilburg/Durbuy.

Stegmann, R., & Hinrichs, E. (1998). *Stylebook for the german treebank in Verbmobil* (Verbmobil Report). Universität Tübingen, Germany: Seminar für Sprachwissenschaft.

Steiner, P. (1995). Anforderungen und Probleme beim Taggen deutscher Zeitungstexte. In H. Feldweg & E. Hinrichs (Eds.), *Lexikon und Text.* Niemeyer, Tübingen.

Steinhaus, H. (1957). The problem of estimation. *Annals of Mathematical Statistics, 28*, 633-648.

Tapanainen, P., & Voutilainen, A. (1993). Ambiguity resolution in a reductionistic parser. In *Proceedings of the 6th conference of the european chapter of the ACL.* Utrecht.

Tzoukermann, E., & Radev, D. R. (in press). Use of weighted finite state transducers in part of speech tagging. In A. Kornai (Ed.), *Extended finite-state models of language.* Cambridge University Press.

Uszkoreit, H., Brants, T., Duchier, D., Krenn, B., Konieczny, L., Oepen, S., & Skut, W. (1998). Studien zur performanzorientierten Linguisitk. Aspekte der Relativsatzextraposition im Deutschen. *Kognitionswissenschaft, 7*(3).

Veenstra, J. (1998). Fast NP chunking using memory-based learning techniques. In *Proceedings of the eighth Belgian-Dutch conference on machine learning.* Wageningen.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In *IEEE transactions on information theory* (p. 260-269).

Volk, M., & Schneider, G. (1998). Comparing a statistical and a rule-based tagger for german. In *Proceedings of KONVENS-98* (p. 125-137). Bonn.

Vosse, T., & Kempen, G. (1991). A hybrid model of human sentence prrocessing: Parsing right-branching, center-embedded and cross-serial dependencies. In *Proceedings of the second international workshop on parsing technologies.* Cancun, Mexico.

Voutilainen, A. (1994). Morphological disambiguation. In F. Karlsson, A. Voutilainen, J. Heikkilä, & A. Anttila (Eds.), *Constraint grammar: a language-independent system for parsing unrestricted text.* Berlin and New York: Mouton de Gruyter.

Wahlster, W. (1993). *Verbmobil: Übersetzung von Verhandlungsdialogen* (Verbmobil Verbundvorhaben / Report 1, Juli 1993). Saarbrücken: DFKI.

Wahlster, W. (1997). *Erkennung, Analyse, Transfer, Generierung und Synthese von Spontansprache* (Verbmobil-Report No. 198). Saarbrücken, Germany: Deutsches Forschungszentrum für künstliche Intelligenz.

Wahlster, W., & Tack, W. (1998). *SFB 378: Ressourcenadaptive Kognitive Prozesse* (Vol. 143; Tech. Rep.). Saarbruecken.

Weischedel, R., Meteer, M., Schwarz, R., Ramshaw, L., & Palmucci, J. (1993). Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, *19(2)*, 359-382.

Wothke, K., Weck-Ulm, I., Heinecke, J., Mertineit, O., & Pachunke, T. (1993). *Statistically based automatic tagging of german text corpora with parts-of-speech – some experiments* (Technical Report No. 75.93.02). Heidelberg: IBM Germany.

Appendix A

Tagsets

A.1 Stuttgart-Tübingen-Tagset (Parts-of-Speech)

No.	Tag	Description
1	ADJA	attributives Adjektiv
2	ADJD	adverbiales oder prädikatives Adjektiv
3	ADV	Adverb
4	APPR	Präposition; Zirkumposition links
5	APPRART	Präposition mit Artikel
6	APPO	Postposition
7	APZR	Zirkumposition rechts
8	ART	bestimmter oder unbestimmter Artikel
9	CARD	Kardinalzahl
10	FM	Fremdsprachliches Material
11	ITJ	Interjektion
12	KOUI	unterordnende Konjunktion mit zu und Infinitiv
13	KOUS	unterordnende Konjunktion mit Satz
14	KON	nebenordnende Konjunktion
15	KOKOM	Vergleichspartikel, ohne Satz
16	NN	normales Nomen
17	NE	Eigennamen
18	PDS	substituierendes Demonstrativpronomen
19	PDAT	attribuierendes Demonstrativpronomen
20	PIS	substituierendes Indefinitpronomen
21	PIAT	attribuierendes Indefinitpronomen
22	PIDAT	attribuierendes Indefinitpronomen mit Determiner
23	PPER	irreflexives Personalpronomen
24	PPOSS	substituierendes Possessivpronomen
25	PPOSAT	attribuierendes Possessivpronomen

No.	Tag	Description
26	PRELS	substituierendes Relativpronomen
27	PRELAT	attribuierendes Relativpronomen
28	PRF	reflexives Personalpronomen
29	PWS	substituierendes Interrogativpronomen
30	PWAT	attribuierendes Interrogativpronomen
31	PWAV	adverbiales Interrogativ- oder Relativpronomen
32	PROAV	Pronominaladverb
33	PTKZU	”zu” vor Infinitiv
34	PTKNEG	Negationspartikel
35	PTKVZ	abgetrennter Verbzusatz
36	PTKANT	Antwortpartikel
37	PTKA	Partikel bei Adjektiv oder Adverb
38	TRUNC	Kompositions-Erstglied
39	VVFIN	finites Verb, voll
40	VVIMP	Imperativ, voll
41	VVINF	Infinitiv, voll
42	VVIZU	Infinitiv mit ”zu”, voll
43	VVPP	Partizip Perfekt, voll
44	VAFIN	finites Verb, aux
45	VAIMP	Imperativ, aux
46	VAINF	Infinitiv, aux
47	VAPP	Partizip Perfekt, aux
48	VMFIN	finites Verb, modal
49	VMINF	Infinitiv, modal
50	VMPP	Partizip Perfekt, modal
51	XY	Nichtwort, Sonderzeichen
52	$,	Komma
53	$.	Satzbeendende Interpunktion
54	$(	sonstige Satzzeichen; satzintern

A.2 NEGRA Corpus

A.2.1 NEGRA Corpus – Phrase Categories

No.	Tag	Description
1	NP	noun phrase
2	AP	adjektive phrase
3	PP	adpositional phrase
4	S	sentence
5	VP	verb phrase (non-finite)
6	VZ	zu-marked infinitive
7	CO	coordination
8	AVP	adverbial phrase
9	AA	superlative phrase with "am"
10	CNP	coordinated noun phrase
11	CAP	coordinated adjektive phrase
12	CPP	coordinated adpositional phrase
13	CS	coordinated sentence
14	CVP	coordinated verb phrase (non-finite)
15	CVZ	coordinated zu-marked infinitive
16	CAVP	coordinated adverbial phrase
17	MPN	multi-word proper noun
18	NM	multi-token number
19	CAC	coordinated adposition
20	CH	chunk
21	MTA	multi-token adjective
22	CCP	coordinated complementiser
23	DL	discourse level constituent
24	ISU	idiosyncratis unit
25	QL	quasi-languag

A.2.2 NEGRA Corpus – Grammatical Functions

No.	Tag	Description
1	AC	adpositional case marker
2	ADC	adjective component
3	AMS	measure argument of adj
4	APP	apposition
5	AVC	adverbial phrase component
6	CC	comparative complement
7	CD	coordinating conjunction
8	CJ	conjunct
9	CM	comparative concjunction
10	CP	complementizer
11	DA	dative
12	DH	discourse-level head
13	DH	discourse-level head
14	DM	discourse marker
15	GL	prenominal genitive
16	GR	postnominal genitive
17	HD	head
18	JU	junctor
19	MC	comitative
20	MI	instrumental
21	ML	locative
22	MNR	postnominal modifier
23	MO	modifier
24	MR	rhetorical modifier
25	MW	way (directional modifier)
26	NG	negation
27	NK	noun kernel modifier
28	NMC	numerical component
29	OA	accusative object
30	OA2	second accusative object
31	OC	clausal object
32	OG	genitive object
33	PD	predicate
34	PG	pseudo-genitive
35	PH	placeholder

No.	Tag	Description
36	PM	morphological particle
37	PNC	proper noun component
38	RC	relative clause
39	RE	repeated element
40	RS	reported speech
41	RS	reported speech
42	SB	subject
43	SBP	passivised subject (PP)
44	SP	subject or predicate
45	SVP	separable verb prefix
46	UC	(idiosyncratic) unit component
47	VO	vocative

A.3 Penn Treebank

A.3.1 Penn Treebank – Part-of-Speech Tags

No.	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

A.3.2 Penn Treebank – Phrase Categories

No.	Tag	Description
48	ADJP	Adjective Phrase. *outrageously expensive.*
49	ADVP	Adverb Phrase. *rather timidly.*
50	CONJP	Conjunction Phrase. *as well as.*
51	FRAG	Fragment.
52	INTJ	Interjection.
53	LST	List marker. Includes surrounding punctuation.
54	NAC	Not A Constituent;
55	NP	Noun Phrase.
56	NX	Used within certain complex noun phrases to mark the head of the noun phrase.
57	PP	Prepositional Phrase.
58	PRN	Parenthetical.
59	PRT	Particle.
60	QP	Quantifier Phrase (i.e., complex measure/amount phrase).
61	RRC	Reduced Relative Clause.
62	S	Simple declarative clause.
63	SBAR	Clause introduced by a (possibly empty) subordinating conjunction.
64	SBARQ	Direct question introduced by a *wh*-word or *wh*-phrase.
65	SINV	Inverted declarative sentence, i.e., one in which the subject follows the tensed verb or modal.
66	SQ	Inverted yes/no question, or main clause of a *wh*-question, following the *wh*-phrase in SBARQ.
67	UCP	Unlike Coordinated Phrase.
68	VP	Verb Phrase.
69	WHADJP	*Wh*-adjective Phrase. *how hot.*
70	WHADVP	*Wh*-adverb Phrase.
71	WHNP	*Wh*-noun Phrase. *which book.*
72	WHPP	*Wh*-prepositional Phrase. *of which.*
73	X	Unknown, uncertain, or unbracketable.

A.3.3 Penn Treebank – Function Tags

No.	Tag	Description
1	-ADV (adverbial)	marks a constituent other than ADVP or PP when it is used adverbially
2	-NOM (nominal)	marks free ("headless") relatives and gerunds when they act nominally.
3	-DTV (dative)	marks the dative object in the unshifted form of the double object construction.
4	-LGS (logical subject)	is used to mark the logical subject in passives.
5	-PRD (predicate)	marks any predicate that is not VP.
6	-PUT	marks the locative complement of *put*.
7	-SBJ (surface subject)	marks the structural surface subject of both matrix and embedded clauses, including those with null subjects.
8	-TPC ("topicalized")	marks elements that appear before the subject in a declarative sentence.
9	-VOC (vocative)	marks nouns of address, regardless of their position in the sentence.
10	-BNF (benefactive)	marks the beneficiary of an action (attaches to NP or PP).
11	-DIR (direction)	marks adverbials that answer the questions "from where?" and "to where?"
12	-EXT (extent)	marks adverbial phrases that describe the spatial extent of an activity.
13	-LOC (locative)	marks adverbials that indicate place/setting of the event.
14	-MNR (manner)	marks adverbials that indicate manner, including instrument phrases.
15	-PRP (purpose or reason)	marks purpose or reason clauses and PPs.
16	-TMP (temporal)	marks temporal or aspectual adverbials.
17	-CLR (closely related)	marks constituents that occupy some middle ground between argument and adjunct of the verb phrase.
18	-CLF (cleft)	marks it-clefts ("true" clefts).
19	-HLN (headline)	marks headlines and datelines.
20	-TTL (title)	is attached to the top node of a title when this title appears inside running text.