Statistical Parsing for German

Modeling syntactic properties and annotation differences

Amit Dubey

Dissertation zur Erlangung des Grades eines
Doktors der Philosophie der Philosophischen
Fakultäten der Universität des Saarlandes

Abstract

Statistical parsing research can be described as being *anglo-centric*: new models are first proposed for English parsing, and only then tested in other languages. Indeed, a standard approach to parsing with new treebanks is to adapt fully developed English parsing models to the other language. In this dissertation, however, we claim that many assumptions of English parsing do not generalize to other languages and treebanks because of linguistic and annotation differences. For example, we show that lexicalized models originally proposed for English parsing generalize poorly to German. Even after modifying the models to account for annotation differences, we find the benefit of lexicalization to be far less than in English.

With this as a starting point, we take a closer look what effect that linguistic differences between English and German have on statistical parsing results. We find that a number of linguistic elements of German play a more crucial role than lexicalization. For example, adding a relatively simple model of the German case system to parser accounts for more ambiguity than a complex model including lexicalization. Further studies show that lexical category ambiguity accounts for a surprising amount of parsing mistakes, and while a model of morphology we develop gives mixed results, an error analysis suggets that a correct model of morphology would help with resolving harmful and common verb/adjective ambiguities. In addition, we offer a preliminary model of long-distance dependencies, showing this model helps greatly in resolving ambiguities caused by German free word order constructions.

We also find that the choice of evaluation metric can have a profound impact on parsing performance: it appears that lexicalized models perform better on dependency-based metrics whereas unlexicalized models perform better on labelled bracketing metrics. Other seemingly arbitrary choices also affect parsing results: the choice of search and smoothing algorithm can potentially obscure helpful linguistic disambiguation cues.

The best performing model we develop sets the state-of-the-art performance on the NEGRA and TIGER corpora, with labelled bracketing scores of 76.2 on NEGRA and 79.5 on TIGER. Furthermore, this parser scores 84.0 on dependencies on the NEGRA corpus, also the best reported performance on that corpus, and 86.2 on the TIGER corpus.

Zusammenfassung

Die bisherige Forschung im Bereich des statistischen Parsing ist weitgehend *anglozentrisch*: neue Modelle werden in der Regel zuerst für das Englische vorgeschlagen und erst dann für andere Sprachen getestet. Parser für neue Baumbanken werden üblicherweise nicht neu entwickelt, sondern es wird lediglich ein Parsingmodell für das Englische auf die neue Sprache angepasst (z.B. Beil et al., 1999; Collins et al., 1999; Bikel und Chiang, 2000). In dieser Dissertation wird gezeigt, dass viele der Annahmen, die für das Parsing des Englischen gemacht werden, sich nicht ohne Weiteres auf andere Sprachen und Baumbanken übertragen lassen. Die Gründ dafür sind Unterschiede in der linguistischen Struktur und den Annotationschemata der Baumbanken. Insbesondere zeigen wir, dass lexikalisierte Parsingmodelle, die ursprünglich für das Englische vorgeschlagen wurden, sich nicht gut auf das Deutsche übertragen lassen. Selbst wenn die Modelle abgeändert werden, um Unterschieden in der Annotation Rechnung zu tragen, sind die Leistungsgewinne durch Lexikalisierung im Deutschen deutlich geringer als im Englischen.

Dieses Ergebnis dient uns als Ausgangspunkt für eine weitreichende Untersuchung der Rolle, die die linguistischen Unterschiede zwischen den beiden Sprachen beim statistischen Parsing spielen. Unsere Ergebnisse zeigen, dass die Berücksichtigung von linguistischen Eigenschaften des Deutschen weit wichtiger als Lexikalisierung sind. Zum Beispiel stellt sich heraus, dass ein relativ einfaches Modell des deutschen Kasussystems sich besser zur Bewältigung von Ambiguitäten eignet als ein lexikalisiertes Modell. Weitere Untersuchungen zeigen außerdem, dass die Ambiguität der lexikalischen Kategorien im Deutschen für eine beträchtliche Anzahl von Parsingfehlern verantwortlich ist. Wir schlagen daraufhin ein Morphologiemodell vor, das aber nur eine unzureichende Verbesserung der Parsingleistung vorweisen kann. Eine Fehleranalyse zeigt jedoch, dass ein ideales Morphologiemodell die Parsingleistung deutlich verbessern würde, da es die häufig auftretende Verb/Adjektiv-Ambiguität auflösen könnte. Des weiteren schlagen wir ein Modell von langen Abhängigkeiten vor und zeigen, dass dieses Modell die Auflösung von Wortstellungambiguitäten im Deutschen deutlich verbessert.

Wir konstatieren auch, dass die verwendete Evaluationsmetrik die Parsingleistung wesentlich beeinflusst: Lexikalisierte Modelle erzielen eine deutlich bessere Leistung, wenn eine Dependenzmetrik angewandt wird. Unlexikalisierte Modelle dagegen erzielen eine bessere Leistung unter Verwendung einer Konstitutentenmetrik. Andere Faktoren scheinen darüberhinaus einen Einfluss auf die Parsingleistung zu haben: je nach verwendetem Suchalgorithmus oder Glättungsschema kommen potentiell wichtige Disambiguierungsmerkmale nicht zur Geltung, und die Leistung des Modells fällt ab.

Das beste in dieser Dissertation entwickelte Modell erzielt eine Parsingleistung, die bisher auf dem NEGRA- und TIGER-Korpus unerreicht ist. Das Modell erzielt eine Konstituenten-F-Metrik von 76.2 auf NEGRA und 79.5 auf TIGER. Desweiteren erzielt es eine Dependenz-F-Metrik von 84.0 für NEGRA und 86.2 für TIGER.

Gliederung

Im Weiteren fassen wir den Inhalt dieser Dissertation kurz zusammen. Die wichtigsten Ergebnisse werden in den Kapiteln 3, 4, 5 und 6 vorgestellt. Diese Kapitel beschreiben die Entwicklung und Evaluation unserer Parsingmodelle für das Deutsche. Das Hauptaugenmerk liegt dabei auf den Auswirkungen, die die Syntax des Deutschen und die Annotation der deutschen Baumbanken auf die Parsingleistung haben.

Kapitel 2 Dieses Kapitel führt den Hintergrund für die vorliegende Dissertation ein. Wir geben eine Übersicht über die verwendete Notation und stellen die Konzepte dar, die dem statistischen Parsing zu Grunde liegen. Wir bieten auch einen Überblick über die Literatur zum statistischen Parsing, wobei das Augenmerk auf dem Parsing von nicht-anglophonen Sprachen liegt. Das Kapitel schließt mit einer Diskussion von methodischen Fragestellungen (z.B. Training und Evaluation des Parsers).

Kapitel 3 Der experimentelle Teil der Dissertation beginnt in diesem Kapitel mit einer Darstellung der Ergebnisse, die wir mit einer Reihe von etablierten Parsingmodellen für das Deutsche erzielt haben. Insbesondere testen wir ein unlexikalisierte Basismodell und die lexikalisierten Modelle von Collins (1997) und Charniak (1997). Die beiden lexikalisierten Modelle wurden ursprünglich für das Englische entwickelt, jedoch wurde das Collins-Modell in abgewandelter Form für das Tschechische (Collins et al., 1999) und für das Chinesische eingesetzt. Das Charniak-Modell wurde bereits von anderen Autoren für das Deutsche eingesetzt (Beil et al., 1999). Wir stellen weiterhin Ergebnisse vor, die mit einem unlexikalisierten Parser unter Verwendung von grammatischen Funktionen erzielt wurden (siehe Abschnitt 2.4). Insgesamt zeigt unsere Untersuchung, dass das unlexikalisierte Basismodell eine bessere Leistung als die beiden lexikalisierten Modelle erbringt. Eine weitere Leistungssteigerung wird durch die Hinzunahme von grammatischen Funktionen erzielt (obwohl die Abdeckung dann deutlich geringer ist). Ausgehend von einer Fehleranalyse schlagen wir dann das Konzept der Schwester-Kopf-Dependenz vor. Ein Parser mit Schwester-Kopf-Dependenzen erzielt eine bessere Parsingleistung als das Basismodell, wobei jedoch die auf die Lexikalisierung zurückzuführende Verbesserung relativ gering ist. Diese Verbesserung ist auch geringer als diejenige, die durch die Verwendung von grammatischen Funktionen erzielt werden kann.

Kapitel 4 Ausgehend von der Feststellung, dass grammatische Funktionen die Parsingleistung erhöhen können, beschäftigen wir uns in diesem Kapitel ausführlich mit der Rolle von grammatischen Funktionen in unlexikalisierten Parsingmodellen. Wir beginnen mit Experimenten zur Integration eines lexikalischen Taggers in den Parser. Diese Integration hat den Vorteil, das jetzt keine Abdeckungsprobleme mehr auftreten. Außerdem profitiert der Parser dann indirekt von den Konzepten, die sich in der Tagging-Literatur als nützlich erwiesen haben. Es zeigt sich Außerdem , dass die Anwendung von automatischen Transformationen auf grammatische Funktionen zu einer Erhöhung der Parsingleistung führt, nahezu auf das Niveau des Schwester-Kopf-Modells von Kapitel 3. Darüberhinaus integrieren wir ein Glättungsmodell in den Parser, was dessen Leistung über das Niveau des Schwester-Kopf-Modells hinaus verbessert. Dieses Ergebnis zeigt, dass grammatische Funktionen dem Parser Informationen über das Kasussystem des Deutschen zur Verfügung stellen und daher die Parsingleistung verbessern.

Kapitel 5 Wie im vorherigen Kapitel gezeigt, verbessert die Verwendung eines grammatischen Merkmals (grammatische Funktionen) den Parser. Es stellt sich also die Frage, ob die Hinzufügung weiterer grammatischer Merkmale einen weiteren Leistungsgewinn bringt; diese Fragestellung wird im vorliegenden Kapitel angegangen. Wir beschäftigen uns mit zwei unterschiedlichen Merkmalsmengen. Ausgehend von den Ergebnissen in Kapitel 4 schlagen wir zuerst eine Merkmalsmenge vor, die der Morphologie von Nominalphrasen Rechnung trägt. Die zweite Merkmalsmenge dient der Modellierung von langen Abhängigkeiten. Unsere Ergebnisse zeigen, dass die morphologischen Merkmale nicht zu einer Verbesserung der Parsingleistung führen, die langen Abhängigkeiten jedoch sehr wohl.

Kapitel 6 In den Kapiteln 6, 3, 4 und 5 verwendeten wir nur eine Evaluationsmetrik und stellten nur eingeschränkt Fehleranalysen an. Desweiteren trainierten und testeten wir unsere Modelle nur auf dem NEGRA-Korpus. Dieses Kapitel generalisiert diese Ergebnisse, indem es den jeweils besten Parser aus den vorhergehenden Kapiteln in dreierlei Hinsicht evaluiert: Tagging von lexikalischen Kategorien, Tagging von grammatischen Funktionen und Wort-Wort-Dependenzen. Desweiteren testen wir alle Modelle auch auf dem TIGER-Korpus. Schließlich führen wir auch eine detaillierte Fehleranalyse der besten Parsingmodelle durch.

Kapitel 7 Dieses Kapitel beschließt die Dissertation mit einer Reihe von Schlussbemerkungen.

Acknowledgements

I owe many people thanks for their help and guidance while writing this dissertation. Foremost on this list are my advisors, Matthew Crocker and Frank Keller. Their support and comments have been immensely helpful, especially in the all-important final leg. I am also endeared to Mirella Lapata, who was always ready with pep-talks and good suggestions.

I am grateful to gave received insights from many different people at various stages over the past 3-odd years. In particular, I would like to thank John Carroll, Péter Dienes, Andreas Eisele, Karin Müller, Detlef Prescher as well as the many students and visitors who attended the EGK meetings.

The group secretaries, Magdalena Mitova and Claudia Verburg, were a great help with all matters administrative. I would have been completely lost without Claudia's help during the first and last days in Saarbrücken.

I would like to show special gratitude to Malte and Ute Gabsdil, whose friendship since the very start of the program made it possible to ease in to a new life in a new country. Many kudos also to the whole EGK (IGK?) gang, both in Saarbrücken and in Edinburgh, as well as the psycholinguists. I would especially like to thank Kerstin Hadelich, Alissa Melinger and Sabine Schulte im Walde, as well as pseudo-psychos Christian Braun and Greg Gulrajani for many good times as well as helping out in the hard ones.

Of course, I would like to thank my family back in Canada, who I was able to keep in touch with via late-night phone calls ("what *time* is it over there?"). This also goes for friends, Alex, Freida, Jay and Posey: it's always nice to know that people still like hearing your voice.

I know I could not possibly finish if I had to list each person by name; surely there are many more who I would like to mention as well. For all of you, thank you as well.

Finally, I would like to thank the German Science Foundation (DFG) for funding this work.

Table of contents

List of tables

List of figures

Chapter 1
Introduction

THIS THESIS concerns parsing German with statistical models. Parsing is an important component of natural language understanding. Syntactic analysis is often the first step involved in turning text in to a computationally meaningful form. Indeed, parse trees are often the "deepest" form necessary for some approaches to question answering (Echihabi and Marcu, 2003), machine translation (Yamada and Knight, 2001), and automatic speech recognition (Roark, 2001). On a more cognitive level, computational parsers are the basis of several models of human sentence processing (Jurafsky, 1996; Crocker and Brants, 2000).

In theory, doing well on these tasks depends upon being able to do well at parsing. It is tempting to say that statistical models allow one to do well at parsing. This, at least, appears to be the case with English statistical parsers (Bod, 2003; Charniak, 2000). There is one problem, though. The literature on statistical parsing is *anglo-centric:* it primarily focuses on English. Although the limited interest in other languages can be partially ascribed to the lack of suitable data, this is not a complete justification. For example, in the case of German and Czech the availability of requisite data (known as *treebank* corpora) has led to some initial work, but not to extensive evaluation. Indeed, to our knowledge, no broad-coverage stand-alone statistical parsers for German had been developed or evaluated at the time this work commenced.

Nevertheless, there is a small but growing literature on parsing other languages (see Section 2.3.3 for an extensive discussion). Much of the work, however, focuses on adapting highly tuned models originally developed for English to the new languages. These models make particular assumptions about which linguistic elements (or *features*) are useful for statistical models of syntax. It is well known that the assumptions of English parsing models not only depend on English, but a particular English treebank corpus, the Wall Street Journal (WSJ) section of the Penn Treebank. It is therefore surprising that parsing in new languages has simply taken the features found to be useful on the WSJ English treebank as a given starting point, without questioning the underlying corpus- and annotation-specific assumptions. This is not to say that it is wrong to use complicated models from the English parsing literature. Rather, we argue that a more methodological approach is necessary.

The purpose of this dissertation is to explicitly test many of the assumptions of WSJ parsing in another language. We pick German because its syntax is different from English in ways which challenge the assumptions made in English parsing (see Section 1.1 below). Moreover, the syntactic differences between German and English compelled treebank designers to adopt different annotation styles in German treebank corpora vis à vis the WSJ corpus (Skut et al., 1997). Therefore, using German corpora entails coping with changes in annotation as well as language.

The primary features we test in this new setting are those which have already found to be useful for English parsing. Other than syntactic categories themselves, one of the most common features used in English parsing is lexicalization, i.e. projecting lexical heads on to tree nodes (Magerman, 1995). Another set of features are derived by enriching the nonterminal vocabulary to account for greater context (Johnson, 1998). But perhaps just as interesting are the features which are commonly not used for parse selection, including grammatical functions (cf. Blaheta and Charniak, 2000), and information about non-local dependencies (cf. Dienes, 2004).

The primary goal is to evaluate the effect of various linguistic features on parsing performance. These features consist of both those which have been proven successful in English parsing, as well as those which are available in German treebank corpora. The underlying hypothesis is that the syntactic properties of German, in particular case and word order, affect the relevance and usefulness of linguistic features for parse disambiguation. When stated explicitly, this hypothesis is seemingly uncontroversial. Yet this hypothesis is interestign to test precisely because it has not been stated or tested explicitly in previous work. An additional goal is to build an accurate broad coverage parser for German.

1.1 German Syntax

When questioning the assumptions of statistical parsing for English, it is important to determine which assumptions might be invalidated in German. Not all syntactic differences between German and English necessarily have an impact. For example, a notable aspect of German is the behaviour of particle verbs. A verb like *aufessen* (“eat up”) has the particle in front in the infinitival and past participle, but the particle sits at the end of the verb phrase in (for example) the present tense, as shown in Example 1.1.

Example 1.1.

Er	*isst*	*immer*	*die*	*Wurst*	*auf*
He	eats	always	the	sausage	up

He always eats up the sausage

However, the particles can also occupy the last position of a VP in English. For example, if “the sausage” from the English gloss in Example 1.1 were pronominalized, we would get the sentence: “He always eats it up.” This is not to say that the behaviour is equivalent in the two languages. Rather, we argue there are enough similarities to have confidance that a statistical model which learns the behaviour in English ought to be able to learn the behaviour in German. This is not necessarily the case with two other aspects of German syntax, more variable word order and more productive morphology.

While there is a difference between the two languages, both similar enough that we may discount particle verbs as being an important impact on parsing performance.

There are other aspects of German which we hypothesize are more important. The two which we hypothesize are most prominent are German’s more variable word order and its more productive morphology. English parsers generally assume dependants are local in nature and that syntatic roles may be derived from positional information, both of which are challenged by variable word order. Furthermore, English parsers, especially lexicalized parsers, make strong assumptions about the distribution of words, which in turn depends on the relatively weak morphology of English. Let us look at each of these in more detail, and then discuss why they may cause problems for statistical parsers.

1.1.1 Word Order

In English as in German, word order is strongly influenced by sentence type. There are four main types of sentence ordering: declarative main clauses, declarative subordinate clauses, questions and commands. Declarative main clauses are the most common type, and as in English, the word order in such clauses is normally subject-verb-object (SVO). Example 4.1 shows such a sentence and its gloss in English.

Example 1.2.

Heroische	*Bürokraten*	*verhindern*	*die*	*Verletzung*	*der*	*Regeln*
Heroic	bureaucrats	prevent	the	breach	of	regulations

Unlike English, as Example 4.2 shows, declarative subordinate clauses have a subject-object-verb order (SOV).

Example 1.3.

weil	*Heroische*	*Bürokraten*	*die*	*Verletzung*	*der*	*Regeln*	*verhindern*
because	heroic	bureaucrats	the	breach	of	regulations	prevent

"because heroic bureaucrats prevent the breach of regulations"

The word order in questions (*Verhindern heroische Bürokraten die Verletzung der Regeln?*, "Do heroic bureaucrats prevent the breach of regulations?") and commands (*Verhindert die Verletzung der Regeln!*, "Prevent the breach of regulations!") is largely the same as in English. In English, the subject, verb and objects normally reside in a fixed order, although the position of modifiers are more relaxed (the sentences *Heroically, the bureaucrat prevented the breach of regulations* and *The bureaucrat, heroically, prevented the breach of regulations* and *The bureaucrat prevented the breach of regulations heroically* are all grammatical and have similar meanings). While the syntactic context determines the verb position in German, subjects, objects as well as modifiers have more freedom in their position in the sentence. The position is often determined by constraints such as pronominalization, topicalization, information structure, definiteness and animacy (Uszkoreit, 1987).

In fixed word order languages, the function of a constituent in a sentence is determined by its position or by the use of prepositions. For example, the first constituent in English is expected to be the grammatical subject. This is not always possible when the position of complements is variable, as in German. In many instances, the *case* of a constituent must be used to determine the function. For example, subjects demand the nominative case, but need not occupy the first position in a sentence. Case is marked by the use of determiners and word endings, which brings us to the second of the major differences between German and English: morphology.

1.1.2 Morphology

In many languages, English included, some syntactic properties are realized in morphology: up to exceptions, plurals are formed by adding an -s, past tense is formed by adding -ed, etc. These are present in German, but on the whole, there are more morphological cues for syntactic phenomena in German than in English.

As noted above, case markings play an important role in disambiguating syntactic functions. Case, together with gender and number, play a much more active part in noun phrase declension in German than English. As in English, German pronouns are marked for case (i.e. *er* "he" versus *ihn* "him"). However, in German case also influences the choice of determiner, the endings of adjectives, and, in some cases, the ending of nouns: compare the nominative *der protzige Club* "the swanky club" with the genitive *des protzigen Clubs* "of the swanky club". German has three genders: masculine, feminine and neuter. Unlike English, which only assigns gender to personal pronouns and possessive determiners (e.g. *he/she* and *his/her*), German assigns gender to all nouns. Gender is not only marked in pronouns and determiners but also adjective and noun affixes (e.g. the *-chen* noun suffix is one possible indication of the neuter gender, as is *-in* for feminine). Likewise, number also marked on all lexical components of noun phrases. The markings for gender, number and case are all ambiguous. For example, Table 1.1 shows the suffixes used to decline so-called *strong* adjectives (used when no determiner is present): there are 24 possible combinations, but only six unique forms.

Of course, inflectional morphology does not affect noun phrases alone. Verbs are marked for person and number agreement. As with nouns, there are more forms than in English. For example, English has only two forms for the present tense of "to sleep": sleep and sleeps; German has four.

		Masculine	Feminine	Neuter
Singular	Nominative	-er	-e	-es
	Genitive	-es	-er	-s
	Dative	-em	-er	-em
	Accusative	-en	-e	-es
Plural	Nominative	-e	-e	-e
	Genitive	-er	-er	-er
	Dative	-en	-en	-en
	Accusative	-e	-e	-e

Table 1.1. Declension of strong adjectives

1.1.3 The Effect of German Syntax on Parsing

As we will see later in this dissertation, the differences in word order and morphological productivity between English and German have a profound impact on parsing performance. Why is this so? Consider the case of word order first.

Sentences exhibiting scrambled word order are often analyzed with the use of long-distance dependencies. Long-distance dependencies pose problems for parsers which rely on local information. This is especially pertinent to statistical models, which tend to exclusively use local cues for disambiguation. Long-distance dependencies are not the only choice available to analyze non-standard word orderings. Another approach is to use flatter trees. The central idea behind using flat trees is the parent of a node is less likely to change even though the node does not occupy its normal position. For this approach work, nodes need to be annotated with their relation to the parent. Such grammatical relations have not been a major component of treebank-trained parsers. Even the strategy of assuming flatter tree representation requires some long-distance dependencies: the scrambling need not be directly below a single parent. Two key concepts required to handle word order flexibility, long-distance dependencies and grammatical functions, are not part of standard statistical parsing models. It is therefore unclear how well these models cope with scrambled word order.

Morphology, too, affects parsing in a number of different ways. First, morphological inflections act as cues to help disambiguate certain structures. Indeed, when constituents do not reside in their normal order, morphological information often necessary to resolve the actual grammatical functions of the constituents. Morphological productivity also affects the distribution of word forms. This, in turn, has an effect on how lexicalization works.

1.2 Results

Over the course of this dissertation, we develop models which take German word order and morphology into account, as well as the results of the analysis of features used in the statistical models. We show that these have a strong impact on parsing performance, and allow us to develop parsing models with the highest results for German parsing known to us. Several of the models we develop are purely investigative, and could not be used to parse free text. Most of the models, however, are suitable to be used with any application which requires a syntactic analysis of German. Indeed, some models developed in Chapter 4 are currently being used for tasks such as machine translation, text-to-text generation, and research in semantic similarity.

There are two common ways to evaluate parsing accuracy: labelled bracketing (explained in Section 2.5.2) and dependencies (Section 6.3). The best performing model from Chapter 4 achieves a labelled bracketing score of 76.2, and a dependency score of 84.0 when using a 350,000 word corpus of German newspaper text. On a larger 800,000 word corpus, the same model achieves a labelled bracket score of 79.5, and a dependency score of 86.2. Furthermore, the best performing model of Chapter 3 further achieves a labelled bracketing score 77.4 and a dependency score of 86.6. All numbers are on sentences of 40 words or less. These are the best reported results for broad-coverage German parsing known to us.

1.3 Outline of the Thesis

The bulk of the dissertation is comprised of Chapters 3, 4, 5 and 6, which describe the development and evaluation of the German statistical parsing models. The overall focus is to examine the effect of German syntax, and the effect of the treebank annotations which account for German syntax, on parsing performance.

Chapter 2 We begin by covering background information in Chapter 2. This chapter opens with a review of the notation we use throughout the thesis, before moving on to a description of the underlying ideas behind probabilistic parsing. It then turns to a survey of the literature on probabilistic parsing, with a particular emphasis on parsing in 'new' languages. The chapter ends with a discussion of methodological issues, including how we train and evaluate our parsers.

Chapter 3 As a starting point for the empirical portion of the dissertation, Chapter 3 reports results on several well known parsing models, including an unlexicalized baseline and the lexicalized models of Collins (1997) and Charniak (1997). While both the lexicalized models were developed for English, a modified version of the Collins model has been used for parsing languages as diverse as Czech (Collins et al., 1999) and Chinese, and the Charniak model has been previously used for German (Beil et al., 1999). Results are also reported for an unlexicalized parser augmented with grammatical function tags (cf. Section 2.4). Surprisingly, we find that the unlexicalized baseline parser does better than both lexicalized parsers. Although the coverage is quite low, the unlexicalized parser with grammatical function tags does even better. Following an error analysis, we introduce the concept of *sister-head* lexical dependencies. A parser using sister-head dependencies is able to outperform the unlexicalized baseline, although the improvement due to lexicalization is quite small. Indeed, it is smaller than the improvement due to the use of grammatical functions.

Chapter 4 Seeing that using grammatical functions actually leads to quite accurate parsing, we return to unlexicalized parsing with grammatical functions in Chapter 4. This chapter begins by investigating the integration of a part-of-speech tagger into the parser. This integration eliminates coverage issues, and provides the additional benefit of incorporating advanced and useful concepts from the part-of-speech tagging literature. We find that applying several automatic transformations to the grammatical functions leads to highly accurate parses, nearly competitive with the sister-head model of Chapter 3. After adding smoothing to the parsing model, the parser in fact performs better than the sister-head model. The (transformed) grammatical functions improve accuracy by giving the parser information about German's case system.

Chapter 5 If adding one attribute (grammatical function labels) to the grammar improves parsing performance, would other linguistically motivated attributes help? This is the primary question which motivates Chapter 5. Two different sets of attributes are proposed in this chapter. Based on the success of modelling case in Chapter 4, the first set of features concern noun phrase morphology. The second set is designed to model long-distance dependences. We find that, for our particular model, the morphological features were not helpful, but the long-distance dependencies were.

Chapter 6 Chapters 3, 4 and 5 all use one evaluation measure, and only contain cursory error analyses. In addition, they only consider models trained and tested on the NEGRA corpus. These problems are resolved in Chapter 6, where the best performing parser from each chapter is evaluated for part-of-speech tagging results, grammatical function tagging and word-word dependencies. In addition, the models are all tested on the TIGER corpus. Finally, we provide an in depth error analysis of our most accurate parsing model.

Chapter 7 Finally, in Chapter 7 we finish with concluding remarks.

Chapter 2
Background

This chapter lays out the major foundations upon which the remainder of the dissertation relies. In Section 2.1, we discuss the basic notation and some fundamental concepts of probability theory. While this is a fairly general treatment, Section 2.2 more specifically describes probabilistic context-free grammars, which lie at the basis of many of the parsing models described in this dissertation.

Section 2.3 offers a survey of the literature on probabilistic parsing. We review work in English, but emphasize research in other languages, in particular in German. The most commonly used corpus for German is the NEGRA treebank, which we discuss in Section 2.4. We also describe the related TIGER treebank. Practical aspects of using the NEGRA and TIGER corpora (such as splitting them into training and test data) along with other methodological issues are covered in Section 2.5.

2.1 Foundations and Notation

2.1.1 Basic Probability Theory

Probability theory allows us to build models in the face of uncertain knowledge about the world.[2.1] Because of the uncertainty caused by ambiguity in language, probability theory has found many uses in computational linguistics.

The possible outcomes of an uncertain situation are known as *elementary events*. The set of all possible outcomes is the set of all elementary events, denoted Ω. Sets of elementary events are simply called *events*. The 'opposite' of an event $E \in \Omega$ is called the complement, $\bar{E}$. $\bar{E}$ is defined by the set complement, $\Omega \backslash E$. The set of all possible events is the power set of Ω (i.e. Ω^{Ω}). While it is possible to only use Ω^{Ω}, it is not always practical or useful to do so. However, if we wish to consider a subset $F \subseteq \Omega^{\Omega}$, then F must satisfy the following conditions:

1. $\Omega \in F$
2. If E is an element of F, then $\bar{E}$ is also in F
3. If $E_1, E_2, ..., E_n$ are events in F, then $\bigcup_{i=1}^{n} F_i \in F$

The most important part of a probability model is the probability function, P, which maps events to probabilities. Such a function is known as a probability density function, or p.d.f.[2.2] P must also obey a number of conditions:

1. $0 \leq P(E) \leq 1$ for all $E \in F$
2. $P(\Omega) = 1$
3. for any sequence of events $E_1, E_2, ...,$ which are all mutually exclusive (that is, $E_i \cup E_j = \emptyset$ for any i,j), then $P(\bigcup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} P(E_i)$

2.1. What follows is a brief overview; a more detailed account of probability can be found in Renyi (1970), Ross (1997) or Stroock (1993).

2.2. Sometimes also called a probability distribution function, or simply "distribution"

In the end, Ω, F and P completely describe a probability model. Formally, we define a probability model as the 3-tuple (Ω, F, P).

2.1.1.1 Random Variables

While events lie at the axiomatic basis of probability theory, it is often easier to express some problems in terms of *random variables*. A random variable X is a function that maps events to another set, usually numbers. For example, an indicator random variable maps events to the set $\{0, 1\}$. This random variable takes the value 0 if the event occurs, 1 otherwise. Notationally, the probability that a random variable X takes the value x is written as $P(X = x)$. If the random variable is clear from the context, we may define $P(x) = P(X = x)$. Two useful functions over a random variable X are the expectation $E(X)$ and the variance $Var(X)$:

$$E(X) = \sum_{x \in F} X \cdot P(X = x)$$

$$Var(X) = E(X^2 - E(X)^2)$$

2.1.1.2 Joint and Conditional Distributions

Random variables need not exist in isolation. We may calculate the probability of two or more random variables having certain outcomes. For example, given two random variables X and Y, we may wish to compute the probability that X takes the value x and that Y takes the value y. This probability is written $P(X = x, Y = y)$. If the random variables are clear from the context, this may be abbreviated as $P(x,\ y)$. In this thesis, we at times take notational liberties and write this as $P(xy)$. The conditional probability $P(X = x|Y = y)$ is defined as:

$$P(X = x|Y = y) = \frac{P(X = x, Y = y)}{P(Y = y)}$$

Informally, the conditional probability of x 'given' y is the probability that $X = x$ given that we known that Y is indeed equal to y.

2.1.1.3 Parameterization

We have yet to examine how to specify the probability density function P. Over time, probability theorists have developed a number of different classes of probability functions. For example, a commonly used class is that of *Gaussian* distributions. Gaussians work over the event space of real numbers, R. The probability that a random variable X drawn from a Gaussian distribution takes a value $x \in R$ is defined as:

$$P(X = x) \;=\; \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.1}$$

The two additional numbers σ and μ in equation 2.1 are known as the *parameters* of the distribution[2.3]. The Gaussian p.d.f. works over a countably infinite event space. When using a p.d.f over a finite event space with a distribution which is difficult to describe, we may use a *parameterless* distribution, which assigns one parameter to each event:

$$P(X = x) \;=\; \theta_x \tag{2.2}$$

A parameterless distribution may also be referred to as a histogram distribution or an empirical distribution. For brevity's sake, we let θ represent a vector containing all the θ_x's. That is, if all possible values of X are $x_1, x_2, ...x_n$ then $\theta = (\theta_{x_1}, \theta_{x_2}, ..., \theta_{x_n})$. Because the assignment of probabilities depends upon θ, we may update Equation 2.2 to the following:

$$P(X = x|\theta) \;=\; \theta_x$$

2.3. In this case, the parameters σ and μ happen to be the variance and mean, respectively, although this does not concern us here.

When choosing a probability model, we are faced with two major issues: first, which distribution to pick, and second, given the distribution, what the parameters should be. We will address the first issue in Section 2.2 by looking at probabilistic context free grammars, a distribution useful for parsing; the second we will discuss in Section 2.1.2.

2.1.2 Learning Probability Models

Before we can use a distribution in the form of Equation 2.2 in a practical setting, we need to assign numbers to θ. This is known as finding an *estimate* for θ. *Maximum likelihood* is a common approach to estimation which normally requires access to some training data D. If D is composed of training samples x_0, x_1, ... x_t, and we assume these events to be identically and independently distributed (i.i.d.), then the estimate of θ, known as $\theta^\star$, may be set according to the following formula:

$$\begin{aligned} \theta^\star &= \operatorname*{argmax}_\theta P(D|\theta) \\ &= \operatorname*{argmax}_\theta \prod_{i=0}^{t} P(X=x_i|\theta) \end{aligned} \tag{2.3}$$

We may solve Equation 2.3 for each $\theta^\star_x$. First, we define a count function $\#(\cdot)$:

$$\#(x) = \sum_{i=0}^{t} \delta(x_i = x)$$

This function counts the number of times x occurs in the training data. Then, solving for $\theta^\star_x$, we get:

$$\theta^\star_x = \frac{\#(x)}{t} \tag{2.4}$$

If x is a vector, then Equation 2.4 is the estimator for a joint distribution. If the training data consists of pairs $<x_0, y_0>, <x_1, y_1>, \ldots <x_t, y_t>$ we may similarly construct a conditional distribution $P(X=x|Y=y,\theta)$. As above, we set $P(X=x|Y=y,\theta)=\theta_{x|y}$, and $\theta_{x|y}$ is estimated as:

$$\theta^\star_{x|y} = \frac{\#(x,y)}{\#(y)}$$

2.1.2.1 Sparse Data and Smoothing

Maximum likelihood estimation has a downside: an event E which does not occur in the training data is assigned a probability of zero, and is hence deemed impossible. In reality, E may simply be too infrequent to appear in a small amount of training data rather than being completely impossible. If so, this would be symptom of *sparse data*: the training set is too small to accurately estimate the parameters.

Using a large training set is not always a cure for sparse data. Despite using a training set of 336 million words, Brown et al. (1992) found that 14.7% of word triples on a held-out set did not occur in the training set. *Smoothing* is a more practical solution to sparse data problems. Common approaches to smoothing interpolate between a very specific (and possibly sparse) distribution and a more general distribution which can be estimated more accurately. We can even have multiple levels of generalization, and combine them in a manner such as:

$$\begin{aligned} & P_{\text{smooth}}(w_n|w_{n-1}, w_{n-2}) \\ = {} & \lambda_1 P(w_n|w_{n-1}, w_{n-2}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n) \end{aligned}$$

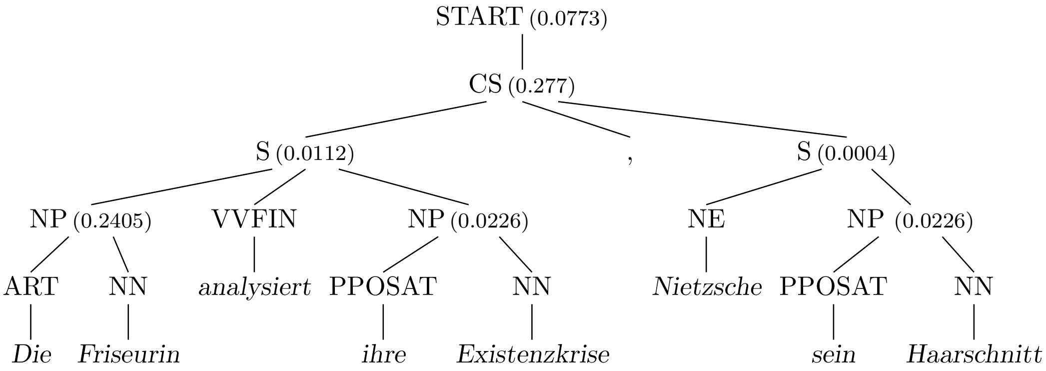


Figure 2.1. A correct parse for Example 2.1 with probabilities shown (see text for gloss).

When using such interpolated smoothing, the probability distributions P are often estimated using the standard maximum likelihood approach. The smoothing parameters (the λ's) require an alternative estimation procedure. Section 4.3 explains three prominent approaches to estimating the smoothing parameters, along with empirical results applied to parsing.

2.2 Probabilistic Context-Free Grammars

A fundamental concept in this thesis is that of probabilistic context-free grammars (Booth and Thompson, 1974), or PCFGs. As the name suggests, PCFGs are a probabilistic version of context-free grammars. We will work with Example 2.1 to illustrate the principals behind PCFGs.

Example 2.1.

Die	*Friseurin*	*analysiert*	*ihre*	*Existenzkrise*	,	*Nietzsche*	*seinen*	*Haarschnitt*
The	hairdresser	analyzes	her	existential crisis	,	Nietzsche	his	hair cut

Figure 2.1 shows a possible parse of this sentence. Conceptually, the parse is derived by continuously applying derivation rules. For example, the first rule applied is START $\rightarrow$ CS, followed by CS $\rightarrow$ S , S. In a PCFG, each rule is associated with a probability. The probability of a rule LHS $\rightarrow$ RHS is $P(\text{RHS} \mid \text{LHS})$. For example, the probability of the rule START $\rightarrow$ CS is $P(\text{CS} \mid \text{START}) = 0.0773$. In Figure 2.1, the probabilities are shown on the parent, e.g. 0.0773 is written next to the START node.

Figure 2.2 shows a second derivation, again with associated probabilities. In the first case the clause *Nietzche seinen Haarschnitt* is considered to be a clausal co-ordinate sister, in the second, it is a noun phrase modifier of *ihre Existenzkrise*. Although both derivations may be licensed by a simple grammar, only the clausal co-ordinate interpretation is correct. Without the use probabilities, it is difficult to pick the first tree over the second. The probabilities of the trees are calculated by multiplying the local rule probabilities. Therefore, the probability of the first tree is 1.178×10^{-11}, and 1.123×10^{-12} for the second. As the first tree has a much higher probability, it is preferred over the second parse.

2.2.1 Lexicalization

In Figures 2.1 and 2.2, nodes associated with part-of-speech (POS) tags do not have probabilities associated with them. In other words, the probability model took the POS tags as a 'certainty', and the input text is essentially ignored. The view as seen by the probability model is essentially that of tree (i) of Figure 2.3.

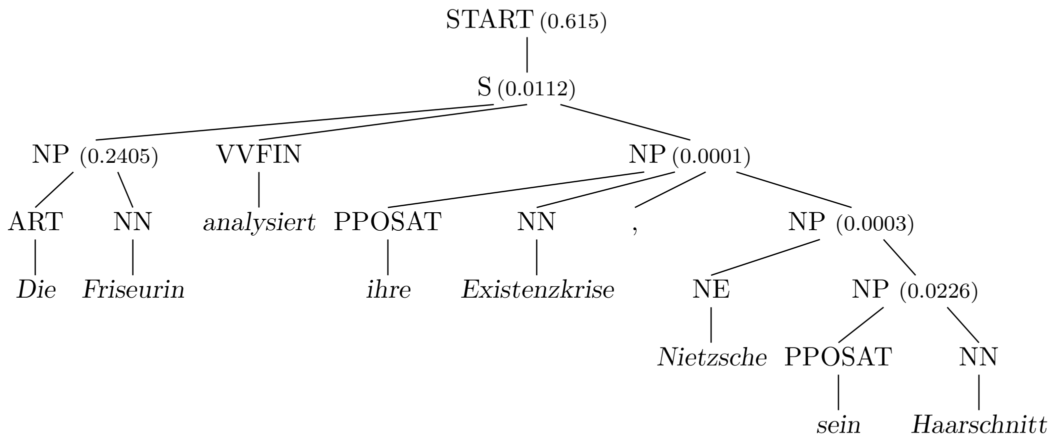
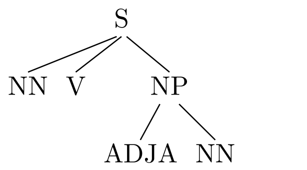
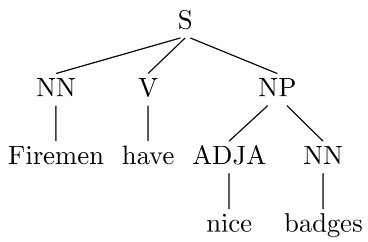


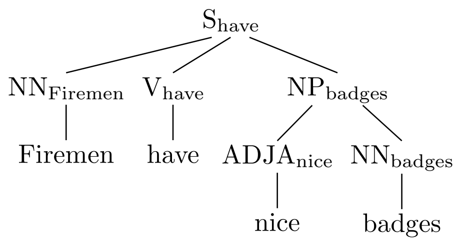
Figure 2.2. An incorrect parse for Example 2.1 with probabilities shown.



i. Unlexicalized Tree



ii. *Partially* Lexicalized Tree



iii. Fully lexicalized Tree

Figure 2.3. Degrees of lexicalization

It is also possible to include word emission probabilities in to the model, by adding rules TAG $\rightarrow$ *word*, and probabilities $P(word|\ \text{TAG})$. In this case, the probability model 'sees' something more like tree (ii) of Figure 2.3. When using such a word emission distribution, it is important to include a special case for unseen words. A common approach for handling unknown words is to create a special word which represents all rare and unseen words.

A simple PCFG was able to pick the right parse from the two possibilities for Example 2.1. PCFGs are not always so successful. Prescher et al. report that only 30% of all sentences are given the right parse using a simple treebank PCFG. Fortunately, treebank PCFGs can be augmented with extra information. A common approach is *lexicalization*, which projects lexical heads on to their parent nodes. Tree (iii) of Figure 2.3 shows a lexicalized tree. Lexicalized models have severe sparse data problems, and therefore require smoothing (see Section 2.1.2.1) and making independence assumptions (see Section 3.1).

2.3 Related Work

2.3.1 Statistical Parsing in English

Lexicalization is an important concept which was introduced fairly early in the history of probabilistic parsing. The concept of using head-head dependencies to lexicalize a grammar is due to Jones and Eisner (1992). Magerman (1995) also uses a lexicalized grammar, deriving the entire grammar from the treebank. Earlier approaches used treebanks for estimating parameters, but used hand-developed grammars (e.g. as in Black et al., 1993). Collins (1996) and Eisner (1996) describe several models for lexicalized parsing with dependency grammar. Among Eisner's dependency models, the best-performing model is the so-called generative model, which is quite similar to the sister-head model we develop in Chapter 3. The primary difference between the sister-head and the generative dependency models is that Eisner's model is much closer to a true dependency grammar: there are no node category labels, just POS tags. The sister-head model, on the other hand, uses both POS tags and syntactic categories. The difference is conceptually minor (e.g. a noun phrase is simply the projection of a noun), but in practice quite different.

Eisner notes that the Collins (1996) and Eisner (1996) introduce complementary concepts. Ideas from both are brought together in Collins (1997). Charniak (1997) proposes a model with an elegant split between 'structural' PCFG probabilities and complex lexical probabilities. Both the Collins (1997) and the Charniak (1997) models are described in detail in Section 3.1.

Charniak (2000) extends the model of Charniak (1997) by introducing a new estimation procedure, making some additional independence assumptions on rule probabilities (similar to Collins, 1997) and adding more contextual information. The contextual information takes the form of grandparent nodes (see Section 4.1.1). Grandparent nodes were first proven useful by Johnson (1998). Klein and Manning (2003), noting that the contextual information found in grandparent nodes did not depend on lexicalization, investigated other non-lexical sources of information which increase parsing accuracy. The result of their investigation is a parser able to parse more accurately than many lexicalized models, including those of Magerman (1995), Collins (1996) and Eisner (1996). We will further discuss accurate unlexicalized models in Chapter 4.

Most of the models discussed above are either based upon PCFGs, or close variants thereof. The key idea behind most of these models is to choose what information is necessary to make *local* parsing decisions. The data-oriented parsing (DOP) approach of Bod (1993) is quite different. DOP-based approaches look at as much information as possible by considering entire subtrees at a time. These subtrees can be arbitrarily large. The standard DOP formulation, due to Bod, is a probability distribution over trees. Others have suggested DOP-inspired models which use discriminative ranking (Collins and Duffy, 2002).

While both DOP and PCFG-based models often require modifications of the Penn Treebank, both lines of research generally adopt grammatical theory underlying the Penn Treebank. This is not the case of all research. Some work aims at disambiguating parses derived from other grammatical theories or formalisms, such as LGF (e.g. Johnson et al., 1999). While we do discuss the parameterizations of some these models in Chapter 5, in this dissertation we primarily focus on grammatical formalisms which closely follow those of the treebank being used.

2.3.2 Statistical Parsing in German

Before discussing statistical parsing in German, it is worth pointing out a topic which is not covered: parsing using formal grammars. The objectives of research in statistical parsing versus that in formal grammar are quite different. Formal grammar is primarily interested in the formal description of linguistic phenomena, whereas in statistical parsing, it is common to take the formal description (the treebank) as given and concentrate on coverage and accuracy. A key issue is frequency: problems interesting in the context of statistical models are those which occur often, whereas problems interesting in the context of formal grammar are those which are especially difficult to describe, even if rare. There are times when the two goals are interwoven: as we shall see in Chapters 4 and 5, adding the kind of information present in formal grammars can increase accuracy. However, the information we use can be seen as pedestrian by the standards of formal grammar. Moreover, the kind of phenomena interesting to formal grammar researchers are either beyond the scope of most statistical grammars, or must be taken as 'given', having been hard-wired into the annotation. For example, in the formal grammar literature, some have argued for a highly structured analysis of verb phrases (Hinrichs and Nakazawa, 1994) while others have argued for a more flat analysis (Nerbonne, 1994). As we shall see in Section 2.4, NEGRA uses flat annotation for verb phrases, making the argument moot.

Returning to the review of statistical methods, we begin with a review of 'shallow' methods of syntactic analysis, such as POS tagging (Schmid, 1995; Brants, 2000) and noun phrase chunking (Scmid and Schulte im Wald, 2000; Skut and Brants, 1998; Brants, 1999).

The Brants POS tagging model (dubbed TnT) is simple, elegant, and achieves near state-of-the-art performance in English, and state-of-the-art performance in German. It does so by using carefully tuned smoothing and an elaborate suffix analyzer. We give a more detailed discussion of the TnT smoothing model in Section 4.3.3, and the suffix analyzer in Section 4.1.3.

Schmid and Schulte im Walde (2000) develop their NP chunking model solely for German. Therefore, they are able to tune their model for German, using auxiliary tools to include information about morphological tagging. Schiehlen (2003) develops a chunker based on dependency grammar. Unlike other chunkers, it works for all constituent types rather than just noun phrases. Noting that heads and dependents tend to be close to one another, Schiehlen's chunker finds dependents occurring within a three-word window of a head. Dependencies outside this window are left for a full parser to find. Brants (1999b) introduces a novel cascaded HMM approach to NP chunking, which achieves prevision and recall of 88.3% and 84.8%. The model can be extended to full parsing Brants(1999a), but results are only reported for an 'interactive' model.

Beil et al. (1999) develop a statistical parser that does not use a treebank grammar. Rather, the parser uses a formally specified grammar. While the grammar performs well enough, it only covers verb-final constructions in German and cannot parse arbitrary sentences. The parameters of the grammar are estimated using the Inside-Outside algorithm (cf. Prescher, 2001). The grammar does have a notion of case, and is able to make use of a morphological tagger to annotate inflectional features. Some inflectional features are collapsed to reduce ambiguity, although it is difficult to assess what impact this has as results are not reported without the reduction. Beil et al. (1999) do not provide an overall test of the statistical grammar's accuracy, but they do test how well the grammar recovers noun chunks and certain kinds of verb dependencies. Their measure for verb dependencies does not take in to account bracketing or word-word dependencies, but rather checks if the category of the dependant is correct. They try a variety of unlexicalized and lexicalized parameterizations, finding that lexicalization only provides a small benefit. The approach of Beil et al. is extended by Schulte im Walde (2000) and Beil et al. (2002). Neither Schulte im Walde nor Beil et al. (2002) test against an annotated held-out test set, instead relying on qualitative and task-based evaluations.

While there is a growing literature on German syntactic analysis using statistical methods, there is surprisingly little work on broad-coverage treebank-trained parsing. One of the first attempts was due to Fissaha (2003). Fissaha et al. report extensive results on the impact of coverage on parsing results, using the NEGRA language as their test corpus. Fissaha et al. do not consider lexicalized models: their primary interest is with unlexicalized models.

Unlexicalized grammars are also the focus of Schiehlen (2004). Following ideas introduced by Klein and Manning (2003) for English parsing, Schiehlen applies automatic transformations to NEGRA which improve parsing accuracy. Among the modifications Schiehlen attempts are copying grammatical functions pertaining to case to the POS tags of articles and all nouns. While case is strongly marked in articles and pronouns, it is only weakly marked in substantive nouns, with common cues only for the genitive singular and dative plural. Furthermore, while case is strongly marked in strong adjectives, Schiehlen does not investigate propogating the tags to strong adjectives.

Dubey and Keller (2003) describe the evaluation well-known lexicalized parsing algorithms on the NEGRA corpus, finding that the models do not appear to generalize well. However, this paper does find that a parser using sister-head dependencies does benefit from lexicalization, although the gain in accuracy is quite small. Levy and Manning (2004) confirm that lexicalization only provides a slight benefit, using a very different model from Dubey and Keller (2003). Levy and Manning also investigate non-local dependencies. This is a topic we cover in Chapter 5, although our results are not directly comparable to Levy and Manning.

2.3.2.1 The Tübingen Corpus and Topological fields

In addition to NEGRA and TIGER, there is a third major syntactically annotated corpus of German, the TüBa-D/Z Tübingen Treebank for German. In addition to Chomskyan syntactic category labels, this treebank also contains annotations for topological field structure (cf. Höhle, 1986). Briefly, topological fields describe restrictions on German word order. Noting that the verb has a fixed position in a sentence, fields are defined in relation to the verb. In a composed tense, there are fields for constituents situated before the finite verb and after the non-finite verb. The so-called *Mittelfeld* (middle field) contains most of a verb's arguments, and lies between the finite and non-finite verbs.

In addition to topological fields, this treebank also annotates syntactic categories, inflectional features, and edge labels. The approach to constituent annotation is different than in NEGRA. Rather than annotating scrambled word orderings with long-distance dependencies, the TüBa-D/Z annotation scheme relies on edge labels. Long-distance dependants are given an edge label which matches that of their long-distance parent. This approach does not always work: sometimes the parent is ambiguous, "too far away", or the daughter ought to have more than one parent. In these cases, a long-distance dependency is explicitly annotated.

There are also some differences concerning local dependencies. For example, unlike NEGRA, prepositional phrases in TüBa-D/Z do have an internal noun phrase (as we shall see in Section 2.4, NEGRA uses a fairly flat annotation scheme).

While the treebank is not yet complete, there has been some initial work on parsing TüBa-D/Z. Ule (2003) reports results on topological field parsing, which fully analyzes topological field boundaries, but does not analyze all syntactic constituents. Ule reports relatively high results: even a PCFG baseline achieves an F-score of 89.6. Related to Ule (2003) is the topologiacal field parser of Becker and Frank (2002). Becker and Frank perform tree transformations to convert the NEGRA corpus into a topological field treebank. Using a modified PCFG grammar derived from this treebank, they find they can recover topological field boundaries with F-scores approaching 93. The results of Ule (2003) and Becker and Frank (2002) together show that topological fields can be found with high accuracy using probabilistic context-free grammars.

It is also possible to do full parsing using TüBa-D/Z annotations. Using a novel memory-based learning (cf. Daelemans et al., 1999) approach on the related TüBa-D treebank, Kübler (2003) reports an F-score of 87.2. The TüBa-D treebank, however, is a corpus of spoken German. Sentences in that treebank are shorter and therefore have less ambiguity than newspaper text copora such as TüBa-D/Z or NEGRA.

2.3.3 Statistical Parsing in Other Languages

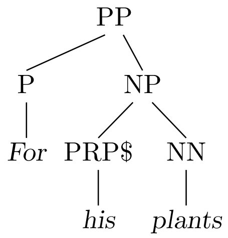
There is a growing literature on parsing other languages. The literature does cover a fair number of languages exhibiting a wide variety of linguistic phenomena. Some of the languages studied, like English and Chinese (Bikel and Chiang, 2000; Levy and Manning, 2003), have fairly rigid rules determining where constituents may appear. German has more relaxed rules concerning the order of constituents compared to English, and Czech (Collins et al., 1999) has yet more relaxed rules than German. Some languages, like English and Chinese, have a fairly weak inflectional morphology. However, other languages, like French (Arun, 2004) and Korean (Lee, 1997) have more productive inflectional rules. Morphological suffixes often have ambiguous meanings in languages like Czech, German and French, whereas in Korean, suffixes unambiguously mark syntactic phenomena.

The Collins (1997) model, which we investigate in Chapter 3, has been tested in several languages, including Czech (Collins et al., 1999) and Chinese (Bikel and Chiang, 2000). The result for Chinese is significantly lower than the performance of the same model for English (see Table 2.1), although the results for French are comparable (Arun, 2004). It is difficult to say how well this model truly fares in Czech as a different evaluation metric is used.

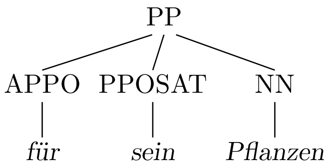
Statistical parsers based on formalisms related to tree adjoining grammar have been applied with some success to Chinese and Korean (Sarkar and Han, 2002), using treebanks with X-bar (Chomsky, 1981) notation. The primary result is that tree adjoining grammar appears to generalize well to Chinese and Korean, albeit requiring a morphological analyzer for the latter. However, it should be noted that Sarkar and Han (2002) do not attempt to analyze long-distance dependencies in Korean. Because Korean is a relatively free word order language, if the grammar does not posit flat structures, then much of the complexity of the grammar will be placed in long-distance dependencies.

There is also a dependency treebank for Korean (Choi, 2001), which does use flatter structures to account for freer word order. Results based on this treebank (e.g. Chung, 2004) have led to highly specialized models for parsing with dependency grammar. These models generally resemble the dependency grammar of Collins (1996), but several aspects have been tuned to Korean. In particular, specialized approaches to measuring the distance of a dependent from its head are used. It is not clear, however, if the success of these specialized distance measures are due to linguistic differences between Korean and English or to annotation differences between dependency-style and X-bar-style treebanks, or both. The flatness of dependency structures may strain the assumption that head-head dependencies are useful for parsing: dependants are much father from their head in dependency-style grammars.

Overall, research in other languages has tended to focus on testing fairly involved models, even though simpler models, such as PCFGs, may often be suitable. Therefore, it is difficult to asses just how much the extra complexity is helping. Moreover, while several free word order languages have been studied, there has been little work in determining just how well models fare with constructions which exhibit non-standard word orderings.



i. Penn Treebank Annotation



ii. NEGRA Annotation

Figure 2.4. There is no PP $\rightarrow$ P NP rule in NEGRA

2.4 Negra and Tiger Annotation

The NEGRA corpus consists of around 350,000 words of German newspaper text with 20,602 sentences. The TIGER corpus is an improper superset of NEGRA, containing about 800,000 words in 40,020 sentences. Both corpora are similarly annotated (with some differences noted below). The annotation scheme (Skut et al., 1997) is modeled to a certain extent on that of the Penn Treebank (Markus et al., 1993), with crucial differences. Most importantly, Negra follows the dependency grammar tradition in assuming *flat syntactic representations*:

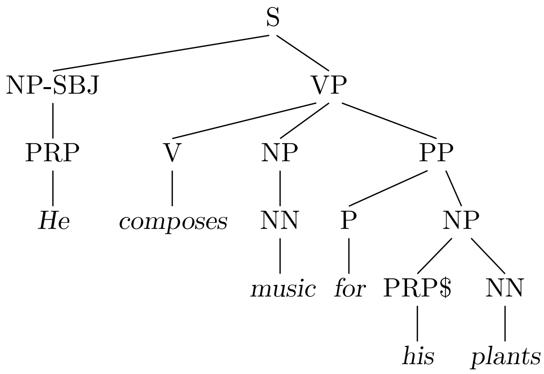
a) There is no PP $\rightarrow$ P NP rule, i.e., the preposition and the noun it selects (and determiners and adjectives, if present) are sisters, dominated by a PP node, as shown in see Figure 2.4. An argument for this representation is that prepositions behave like case markers in German; a preposition and a determiner can merge into a single word (e.g., *in dem* 'in the' becomes *im*).

b) There is no S $\rightarrow$ NP VP rule. Rather, the subject, the verb, and its objects are all sisters of each other, dominated by an S node (see Figure 2.5). This is a way of accounting for the semi-free word order of German (cf. Section 1.1.1): the first NP within an S need not be the subject.

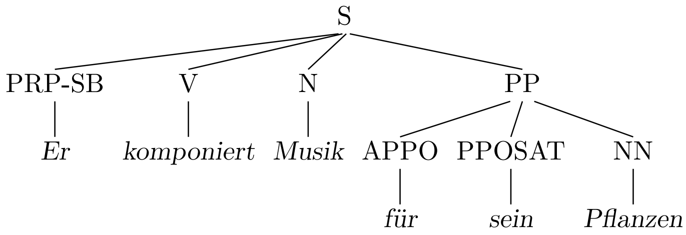
c) There is no SBAR $\rightarrow$ Comp S rule. Main clauses, subordinate clauses, and relative clauses all share the category S in Negra; as shown in Figure 2.6, complementizers and relative pronouns are simply sisters of the verb.

Another idiosyncrasy of Negra is that it assumes special *co-ordinate categories*. A coordinated sentence has the category CS, a coordinate NP has the category CNP, etc. While this does not make the annotation more flat, it substantially increases the number of non-terminal labels. Negra also contains *grammatical function* (GF) labels that augment phrasal and lexical categories. Example are MO (modifier), HD (head), SB (subject), and OC (clausal object).

The TIGER corpus uses GF labels to differentiate PP objects (OP) from PP modifiers. The other major difference between TIGER and NEGRA which concerns us is the choice of label for proper nouns. NEGRA uses the MPN (multiword proper noun) whereas TIGER uses PN (proper noun).



i. Penn Treebank Annotation



ii. NEGRA Annotation

Figure 2.5. There is no S $\rightarrow$ NP VP rule in NEGRA

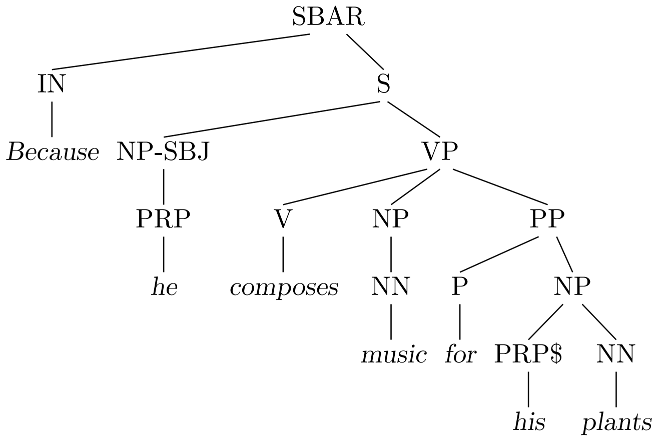
2.5 Methodology

2.5.1 Data

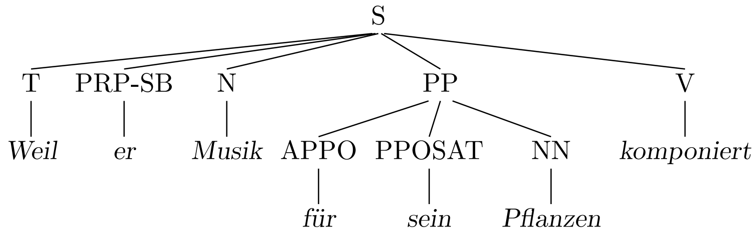
All the experiments in Chapters 3, 4, 5 and some of the experiments in Chapter 6 use the NEGRA corpus. Some experiments in Chapter 6 are performed on the TIGER corpus. All the experiments use the treebank format of these corpora. This format, which is included in the NEGRA and TIGER distributions, is derived from the native format by replacing crossing branches with traces.

The NEGRA corpus consists of 20,602 sentences. The first 18,602 sentences constituted the training set. Of the remaining 2,000 sentences, the first 1,000 served as the test set, and the last 1000 as the development set. To increase parsing efficiency, we removed all sentences with more than 40 words from the test and development sets. This resulted in a test set of 968 sentences and a development set of 975 sentences. Preliminary results were derived on the development set, and the test set remained unseen until all parameters were fixed. The results reported in this thesis were obtained on the test set, unless stated otherwise.

The TIGER corpus consists of 40,002 sentences. Karin Müller (personal communication) suggests a 'standard' split of TIGER into training, testing and development sets for other researchers to follow. The standard split is created by placing each sentence into one of 20 buckets. Numbering from one, the $i - 1^{\text{th}}$ sentence is placed in to bucket number $i \bmod 20$. The test set consists of buckets number 1 to 18, the development set bucket 19, and the test set bucket 20. We adhere to this standard for our experiments in Chapter 6.



i. Penn Treebank Annotation



ii. NEGRA Annotation

Figure 2.6. There is no SBAR $\rightarrow$ Comp S rule in NEGRA

2.5.2 Evaluation

Evaluation is an important part of this dissertation. We use several different evaluation metrics. In earlier parts of the thesis, we use the most common metrics: labelled brackets and crossing brackets (Black et al., 1992; Magerman, 1995). In Chapter 6, we also re-evaluate the particularly interesting models using alternative evaluation measures, including word-word dependencies.

Labeled bracketing scores tend to be reported more often than crossing bracket scores in the literature. It is possible to measure the precision, recall and F-score of labelled bracketing, again the F-score or average is the most common metric to report. We normally report precision, recall and F-score of labelled brackets, as well as the crossing bracket measures. Where space and brevity demand it, we follow convention and report only the F-score of labelled brackets.

When comparing two results, we follow the standard practice in statistical parsing literature, and do not attempt to perform hypothesis testing. This practise is not due to an arbitrary choice. It is not straightforward to construct a decision rule for F-scores: using the naïve approach, an F-score has zero degrees of freedom. It is possible to use *average* F-score, i.e., to calculate the F-score of each sentence, and average the results together. However, average F-scores are not Gaussian distributed in our data: there are two peaks, one close to 1.0, and another peak usually close to 0.6. Therefore, a non-parametric test would be necessary. Unfortunately, another problem arises: averaging the F-scores biases the results, giving smaller sentences a bigger weighting. There are two other solutions: we could use expensive sampling-based hypothesis testing, or use nonparametric tests on precision and recall. However, as our test set is quite large, we expect variance due to sampling to be relatively small.

2.6 Summary

In this chapter, we have discussed issues of notation; reviewed the concept of probabilistic context-free grammars; discussed relevant related work; given an overview of the NEGRA and TIGER annotation schemes; and introduced the methodology we use for experiments in Chapters Chapters 3, 4, 5 and 6. Our review of the related work shows there has been little work on treebank-trained parsing in German, and indeed in other languages as well. What research has been done in other languages often tests complex models originally derived from the English parsing literature in isolation, not testing against simpler baselines such as unlexicalized PCFGs when it is possible. This makes it difficult to judge how much is gained by the extra complexity. Moreover, any complex model necessarily contains features from a wide variety of sources, including non-local structural information as well as lexicalization (Klein and Manning, 2003). There has not as yet been any attempt to investigate, part by part, how the components of more complex models fare in new languages.

Chapter 3
Lexicalized Parsing

We begin by experimenting with lexicalized parsing. This is a logical starting point as the best-performing parsing models for English use some form of lexicalized grammar (Charniak, 1997; Charniak, 2000; Collins, 1997). Indeed, lexicalization has been shown to dramatically increase parsing performance. But does this result hold true for other languages? Because the effect of lexicalization is so strong in English, we may initially assume that lexicalization ought to be useful in German. To test this hypothesis, we compare two lexicalized models with an unlexicalized baseline.

We show that lexicalized parsers behave quite differently in German than in English. We argue there are three reasons for this: (i) scoring effects; (ii) assumptions about annotation; and (iii) assumptions about the distribution of words in English. We introduce a new model which appears to account for the scoring and annotation effects. The third factor, however, gives rise to our main thesis: that new techniques are required to make lexicalization work in languages, such as German, which have a productive morphology.

This chapter is structured as follows. Section 3.1 describes two standard lexicalized models (Carroll and Rooth, 1998; Collins, 1997), as well as an unlexicalized baseline model. Section 3.2 presents a series of experiments that compare the parsing performance of these three models (and several variants) on NEGRA. The results show that both lexicalized models fail to outperform the unlexicalized baseline. This is at odds with what has been reported for English. Learning curves show that the poor performance of the lexicalized models is not due to lack of training data.

An alternative explanation is explored in Section 3.3. An error analysis for the Collins (1997) lexicalized model shows that the head-head dependencies used in this model fail to cope well with the flat structures in NEGRA. We propose an alternative model that uses sister-head dependencies instead. This model outperforms the two original lexicalized models, as well as the unlexicalized baseline.

Section 3.4 compares the performance of the underlying structural models of both the head-head and sister-head parsers. We find the deficiency of the head-head model is not due to lexicalization *per se*, but rather due to poor assumptions of the structural model. Nonetheless, we show that the improvement due to lexicalized is quite a bit smaller than in English. We argue part of the difference in the impact of lexicalization in English and German is due to the very different distribution of words in German.

To better assess the impact of flat structures, in Section 3.5 we 'unflatten' the NEGRA grammar. Although flatness still has a negative effect on the sister-head parser, this section shows the theoretical intuition justifying flatter structures appears to be grounded by our experimental evidence. Overall, Section 3.5 gives evidence that scoring effects have an impact on overall parsing performance.

In Section 3.6, we make some preliminary explorations of how accurate the sister-head parser is on a notably difficult construction in German, the verb-final clause. This section lays the basis upon which we study other German syntactic constructions in Sections 1.1.1. Finally, we offer some concluding remarks in Section 3.7.

This is the extension of joint work done with Frank Keller, previously published as Dubey and Keller (2003). The work in this chapter remains the most exhaustive study of broad-coverage lexicalized probabilistic parsing in German.

3.1 The Models

As we saw in Section 2.2, we may induce a probability distribution over a context-free grammar by assigning each rule LHS $\rightarrow$ RHS an expansion probability $P(\text{RHS} \mid \text{LHS})$. The probabilities for all rules with the same left hand side must to sum to one, and the probability of a parse tree T is defined as the product of the probabilities of all rules applied in generating T.

In a lexicalized grammar, we also generate words on the RHS and use them as part of the LHS context upon which a RHS is conditioned. Let us examine the process in more detail. If P is the LHS of a rule, and if we pick one child to be the head (call it H), and if there are m children to the left of the head and n children to the right, then we may write the rule LHS $\rightarrow$ RHS as:

$$P \rightarrow L_m ... L_1 \, H R_1 ... R_n$$

To keep each child on the RHS visually distinct, we will draw boxes around them:

$$\boxed{P} \rightarrow \boxed{L_m} ... \boxed{L_1} \boxed{H} \boxed{R_1} ... \boxed{R_n}$$

In this more verbose format, the probability of an unlexicalized rule is written as:

$$P_{unlex}(LHS|RHS) = P(\boxed{L_m} ... \boxed{L_1} \boxed{H} \boxed{R_1} ... \boxed{R_n} | \boxed{P})$$

Let C be any daughter. Further, let w_C be the head word of daughter C and t_C the tag of the head word of C. If H is the head daughter of the rule and P is the parent, note that w_H is the same as w_P. Using this encoding, we may write a simple lexicalized rule as:

$$P_{lex}(LHS|RHS) = P\left(\boxed{\begin{array}{c} L_m \\ w_{L_m} \end{array}} ... \boxed{\begin{array}{c} L_1 \\ w_{L_1} \end{array}} \boxed{\begin{array}{c} H \\ \end{array}} \boxed{\begin{array}{c} R_1 \\ w_{R_1} \end{array}} ... \boxed{\begin{array}{c} R_n \\ w_{R_n} \end{array}} \middle\| \boxed{\begin{array}{c} P \\ w_P \end{array}} \right) \tag{3.1}$$

Alternatively, if we chose to generate the POS tags as the same time as the head word, we would get:

$$P_{lex}(LHS|RHS) = P\left(\boxed{\begin{array}{c} L_m \\ t_{L_m} \\ w_{L_m} \end{array}} ... \boxed{\begin{array}{c} L_1 \\ t_{L_1} \\ w_{L_1} \end{array}} \boxed{\begin{array}{c} H \\ \\ \end{array}} \boxed{\begin{array}{c} R_1 \\ t_{R_1} \\ w_{R_1} \end{array}} ... \boxed{\begin{array}{c} R_n \\ t_{R_n} \\ w_{R_n} \end{array}} \middle\| \boxed{\begin{array}{c} P \\ t_P \\ w_P \end{array}} \right) \tag{3.2}$$

The head-lexicalized PCFG model of Carroll and Rooth (1998) uses Equation 3.1 as its basis, while the model proposed by Collins (1997) begins with Equation 3.2 as its basis (we henceforth refer to the models as the *C&R model* and *Collins model*, respectively). Directly estimating the parameters of Equations 3.1 and 3.2 would lead to severe sparse data problems. Both the C&R and Collins models use smoothing and novel independence assumptions to overcome sparse data. Both models use similar approaches to smoothing, so we do not address it here. On the other hand, the models greatly diverge on their independence assumptions, so it is worth taking a closer look at these assumptions.

We examine the C&R model first. Starting with Equation 3.1, the first assumption is to separate the generation of lexical items from the generation of rules.

$$\begin{aligned} P_{lex}(LHS|RHS) &= P\left(\boxed{\begin{array}{c} L_m \\ w_{L_m} \end{array}} ... \boxed{\begin{array}{c} L_1 \\ w_{L_1} \end{array}} \boxed{\begin{array}{c} H \\ \end{array}} \boxed{\begin{array}{c} R_1 \\ w_{R_1} \end{array}} ... \boxed{\begin{array}{c} R_n \\ w_{R_n} \end{array}} \middle\| \boxed{\begin{array}{c} P \\ w_P \end{array}} \right) \\ &= P\left(\boxed{\begin{array}{c} L_m \\ \end{array}} ... \boxed{\begin{array}{c} L_1 \\ \end{array}} \boxed{\begin{array}{c} H \\ \end{array}} \boxed{\begin{array}{c} R_1 \\ \end{array}} ... \boxed{\begin{array}{c} R_n \\ \end{array}} \middle\| \boxed{\begin{array}{c} P \\ w_P \end{array}} \right) \\ &\quad \cdot P\left(\boxed{\begin{array}{c} \\ w_{L_m} \end{array}} ... \boxed{\begin{array}{c} \\ w_{L_1} \end{array}} \boxed{\begin{array}{c} \\ w_{R_1} \end{array}} ... \boxed{\begin{array}{c} \\ w_{R_n} \end{array}} \middle\| \boxed{\begin{array}{c} P \\ w_P \end{array}} \boxed{\begin{array}{c} L_m \\ \end{array}} ... \boxed{\begin{array}{c} R_n \\ \end{array}} \right) \end{aligned}$$

Second, we assume that all the lexical items are generated independently of one another.

$$
\begin{aligned}
&= P\left(\boxed{L_m} \ldots \boxed{L_1}\,\boxed{H}\,\boxed{R_1} \ldots \boxed{R_n} \;\middle\|\; \boxed{\begin{array}{c} P \\ w_P \end{array}} \right) \\
&\quad \cdot \left[\prod_{i=1}^{m} P\left(\boxed{\begin{array}{c} \\ w_{L_i} \end{array}} \;\middle\|\; \boxed{\begin{array}{c} P \\ w_P \end{array}}\,\boxed{\begin{array}{c} L_i \\ \\ \end{array}} \right) \right] \cdot \left[\prod_{i=1}^{n} P\left(\boxed{\begin{array}{c} \\ w_{R_i} \end{array}} \;\middle\|\; \boxed{\begin{array}{c} P \\ w_P \end{array}}\,\boxed{\begin{array}{c} R_i \\ \\ \end{array}} \right) \right]
\end{aligned}
$$

The last line gives us the formulation of the C&R model. Notice that it has a distinctive feature: an almost complete separation of rule probabilities from lexical probabilities. Indeed, because the rule probabilities are almost the same as with a PCFG, the C&R model is a minimal departure from the standard unlexicalized PCFG, which makes it ideal for a direct comparison. Note that the C&R model is essentially the same as that of Charniak (1997); we will nevertheless use the label 'Carroll and Rooth model' as we are using their implementation (see Section 3.2.1).

In contrast to the C&R approach, the Collins model does not compute rule probabilities directly. Rather, they are generated using a Markov process that makes a different set of independence assumptions than the C&R model. Starting with Equation 3.2, the Collins model first assumes that everything to the left and right of the head are generated independently of one another.

$$
\begin{aligned}
P_{lex}(LHS|RHS) &= P\left(\boxed{\begin{array}{c} L_m \\ t_{L_m} \\ w_{L_m} \end{array}} \ldots \boxed{\begin{array}{c} L_1 \\ t_{L_1} \\ w_{L_1} \end{array}}\,\boxed{\begin{array}{c} H \\ \\ \\ \end{array}}\,\boxed{\begin{array}{c} R_1 \\ t_{R_1} \\ w_{R_1} \end{array}} \ldots \boxed{\begin{array}{c} R_n \\ t_{R_n} \\ w_{R_n} \end{array}} \;\middle\|\; \boxed{\begin{array}{c} P \\ t_P \\ w_P \end{array}} \right) \\
&= P\left(\boxed{\begin{array}{c} H \\ \\ \\ \end{array}} \;\middle\|\; \boxed{\begin{array}{c} P \\ t_P \\ w_P \end{array}} \right) \\
&\quad \cdot P\left(\boxed{\begin{array}{c} L_m \\ t_{L_m} \\ w_{L_m} \end{array}} \ldots \boxed{\begin{array}{c} L_1 \\ t_{L_1} \\ w_{L_1} \end{array}} \;\middle\|\; \boxed{\begin{array}{c} P \\ t_P \\ w_P \end{array}}\,\boxed{\begin{array}{c} H \\ \\ \\ \end{array}} \right) \\
&\quad \cdot P\left(\boxed{\begin{array}{c} R_1 \\ t_{R_1} \\ w_{R_1} \end{array}} \ldots \boxed{\begin{array}{c} R_n \\ t_{R_n} \\ w_{R_n} \end{array}} \;\middle\|\; \boxed{\begin{array}{c} P \\ t_P \\ w_P \end{array}}\,\boxed{\begin{array}{c} H \\ \\ \\ \end{array}} \right)
\end{aligned}
$$

Then, we make the 0^{th} order Markov assumption, with the result that each node on the left and right are generated independently of one another:

$$
\begin{aligned}
&= P\left(\boxed{\begin{array}{c} H \\ \\ \\ \end{array}} \;\middle\|\; \boxed{\begin{array}{c} P \\ t_P \\ w_P \end{array}} \right) \\
&\quad \cdot \left[\prod_{i=0}^{m} P\left(\boxed{\begin{array}{c} L_i \\ t_{L_i} \\ w_{L_i} \end{array}}, d(i) \;\middle\|\; \boxed{\begin{array}{c} P \\ t_P \\ w_P \end{array}}\,\boxed{\begin{array}{c} H \\ \\ \\ \end{array}} \right) \right] \\
&\quad \cdot \left[\prod_{i=0}^{n} P\left(\boxed{\begin{array}{c} R_i \\ t_{R_i} \\ w_{R_i} \end{array}}, d(i) \;\middle\|\; \boxed{\begin{array}{c} P \\ t_P \\ w_P \end{array}}\,\boxed{\begin{array}{c} H \\ \\ \\ \end{array}} \right) \right]
\end{aligned}
\tag{3.3}
$$

Notice that this results in a huge loss of information: the ith node no longer depends upon the previous $i-1$ nodes. While the C&R model makes the same assumption for lexical affinities, it does not make this assumption for syntactic categories. To compensate for harmful effects of this assumption, the *distance measure*, $d(i)$, is added to approximate the now-missing nodes. The distance measure consists of two binary numbers, which are set to '1' if the answers to the two following questions are 'yes':

i. Is there a verb between H and the ith constituent?

ii. Is there punctuation between H and the ith constituent?

For details on the distance measures, refer to Collins (1999). The three models presented here, along with the new model we suggest in Section 3.3, will serve as the basis for the experiments in this chapter.

3.2 Parsing with Head-Head Parameters

Having introduced the models, in this section we turn our attention to testing their performance on the NEGRA corpus. The main hypothesis is that the lexicalized models will outperform the unlexicalized baseline. Another prediction is that adding NEGRA-specific information to the models will increase parsing performance. Therefore, we test a variant model that includes grammatical function (GF) labels, i.e., the set of categories was augmented by the function tags specified in NEGRA (see Section 2.4).

Adding grammatical functions is a way of dealing with the word order facts of German (see Section 1.1.1) in the face of NEGRA's very flat annotation scheme. For instance, subject and object NPs have different word order preferences (subjects tend to be preverbal, while objects tend to be postverbal), a fact that is captured if subjects have the label NP-SB, while objects are labelled NP-OA (accusative object), NP-DA (dative object), etc. Also the fact that verb order differs between subordinate and main clauses is captured by the function labels: the former are labelled S, while the latter are labelled S-OC (object clause), S-RC (relative clause), etc.

3.2.1 Method

Data Sets All the experiments reported here use the division of the NEGRA corpus in to training, testing and development sets as described in Section 2.5.1. Early versions of the models were tested on the development set, and the test set remained unseen until all parameters were fixed. The final results reported in this chapter were obtained on the test set, unless stated otherwise. Before applying the models we use here, we first remove all empty nodes. While the parses we discuss here cannot handle empty nodes or any of phenomena that depend on them, we will return to this topic in Chapter 5.

Grammar Induction For the unlexicalized PCFG model (henceforth *baseline model*), we used the probabilistic left-corner parser Lopar (Schmid, 2000). When run in unlexicalized mode, Lopar implements the model described in Section 4.2.1. A grammar and a lexicon for Lopar were read off the NEGRA training set, after removing all grammatical function labels.

The C&R model was again realized using Lopar, which in lexicalized mode implements the model in Section 4.2.2. Lexicalization requires that each rule in a grammar has one of the categories on its right hand side annotated as the head. For the categories S, VP, AP, and AVP, the head is marked in NEGRA. For the other categories, we used the rule listed in Appendix A to determine the head. These head-finding rules were developed by hand, as is standard practise for the Penn Treebank.

	Recall	Precision	F-score	CB	0 CB	$\leqslant 2$ CB	Coverage
Baseline	70.6	66.7	68.6	1.03	58.2	84.5	94.4
Baseline+GF	70.4	65.5	67.9	1.07	58.0	85.0	79.2
C&R	68.0	60.1	63.8	1.31	52.1	79.5	94.4
C&R+GF	67.7	60.3	63.8	1.31	55.7	80.2	79.2
Collins	67.9	66.1	67.0	0.73	65.7	89.5	95.2

Table 3.1. Results with TnT tagging

As an implementation of the Collins parser was not available to us at the time this experiment was done, we used a re-implementation of this model. For training, empty categories were removed from the training data, as the model cannot handle them. The same head finding strategy was applied as for the C&R model.

In this experiment, only head-head statistics were used (see Section 3.2). The original Collins model uses sister-head statistics for non-recursive NPs. This will be discussed in detail in Section 3.3.

Training and Testing We estimated the model parameters using maximum likelihood estimation. Both Lopar and the Collins model use various techniques to smooth the estimates. Lopar uses absolute discounting (Ney et al., 1994) whereas Collins uses a variant of Witten-Bell smoothing (Witten and Bell, 1991). We explore Witten-Bell smoothing, as well as an extension of absolute discounting (Kneser and Ney, 1995) as applied to unlexicalized parsing in Section 4.3. For the details of the smoothing in these lexicalized models, though, the reader is referred to Schmid (2000) and Collins (1997). For the C&R model, we used a cutoff of one for rule frequencies and lexical choice frequencies (the cutoff value was optimized on the development set).

We also tested variants of the baseline model and the C&R model that include grammatical function information, as we hypothesized that this information might help the model to handle word order variation more adequately, as explained above.

Lopar and the Collins model differ in their handling of unknown words. In Lopar, a POS tag distribution for unknown words has to be specified, which is then used to tag unknown words in the test data. The Collins model treats any word seen fewer than five times in the training data as unseen and uses an external POS tagger to tag unknown words. In order to make the models comparable, we used a uniform approach to unknown words. All models were run on POS-tagged input; this input was created by tagging the test set with a separate POS tagger, for both known and unknown words. We used TnT (Brants, 2000), trained on the NEGRA training set. The tagging accuracy was 97.12% on the development set.

In order to obtain an upper bound for the performance of the parsing models, we also ran the parsers on the test set with the correct tags (as specified in NEGRA), again for both known and unknown words. We will refer to this mode as 'perfect tagging'.

All models were evaluated using standard PARSEVAL measures. We report labelled recall (LR) labelled precision (LP), F-score, average crossing brackets (CBs), zero crossing brackets (0CB), and two or less crossing brackets ($\leq$ 2CB). We also give the coverage (Cov), i.e., the percentage of sentences that the parser was able to parse.

3.2.2 Results

The results for all three models and their variants are shown in Table 3.1 for TnT tags and Table 3.2 for perfect tags. The baseline model achieves an F-score of 68.6 with TnT tags. Adding grammatical functions reduces the figure slightly, and makes coverage drop substantially, by about 15%. The C&R model performs worse than the baseline, with an F-score of 63.8 (for TnT tags). Once again, adding grammatical function reduces performance slightly. The Collins models also performs worse than the baseline, with an F-score of 67.0.

	Precision	Recall	F-score	Avg CB	0CB	$\leqslant 2$ CB	Coverage
Baseline	73.0	70.0	71.5	0.88	60.0	87.4	95.3
Baseline+GF	81.1	78.4	79.7	0.46	74.3	95.3	65.4
C&R	70.8	63.4	66.9	1.17	55.0	82.2	95.3
C&R+GF	81.2	76.8	78.9	0.48	73.5	94.2	65.4
Collins	68.6	66.9	67.7	0.71	65.0	89.7	96.2

Table 3.2. Results with perfect tagging

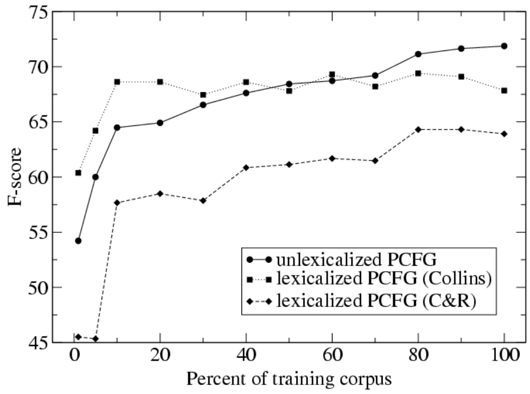


Figure 3.1. Learing curves for all three models

Performance using perfect tags (an upper bound of model performance) is 2--3% higher for the baseline and for the C&R model. The Collins model gains only about 1%. Perfect tagging results in a performance increase of over 10% for the models with grammatical functions. This is not surprising, as the perfect tags (but not the TnT tags) include grammatical function labels. However, we also observe a dramatic reduction in coverage (to about 65%).

3.2.3 Discussion

We added grammatical functions to both the baseline model and the C&R model, as we predicted that this would allow the model to better capture the word order facts of German. However, this prediction was not borne out: performance with grammatical functions (on TnT tags) was slightly worse than without, and coverage dropped substantially. A possible reason for this is sparse data: a grammar augmented with grammatical functions contains many additional categories, which means that many more parameters have to be estimated using the same training set. On the other hand, a performance increase occurs if the perfectly tagged input contains grammatical function labels. Although this comes at the price of an unacceptable reduction in coverage, in Chapter 4 we will examine ways to improve the coverage of a GF-based parser.

The most surprising finding is that the best performance was achieved by the unlexicalized PCFG baseline model. Both lexicalized models (C&R and Collins) performed worse than the baseline. This results is at odds with what has been found for English, where lexicalization is standardly reported to increase performance by about 10%. The poor performance of the lexicalized models could be due to a lack of sufficient training data: our NEGRA training set contains approximately 18,000 sentences, and is therefore significantly smaller than the Penn Treebank training set (about 40,000 sentences). NEGRA sentences are also shorter: they contain, on average, 15 words compared to 22 in the Penn Treebank.

	Penn	NEGRA
NP	2.20	3.08
PP	2.03	2.66
VP	2.32	2.59
S	2.22	4.22

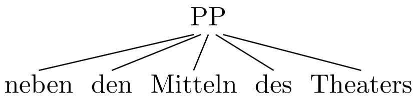
Table 3.3. Average number of daughters of the given categories in the Penn Treebank and NEGRA

We computed learning curves for the unmodified variants (without grammatical functions) of all three models on the development set. The result (see Figure 3.1) shows that there is no evidence for an effect of sparse data. For both the baseline and the C&R model, a fairly high F-score is achieved with only 10% of the training data. A slow increase occurs as more training data is added. The performance of the Collins model is even less affected by training set size. This is probably due to the fact that it does not use rule probabilities directly, but generates rules using a Markov chain.

3.3 Parsing with Sister-Heads

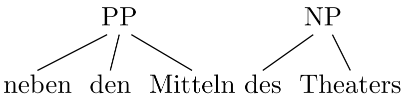
As we saw in the last section, lack of training data is not a plausible explanation for the sub-baseline performance of the lexicalized models. In this section, we therefore investigate an alternative hypothesis: the lexicalized models do not cope well with the flat rules of NEGRA. We will focus on the Collins model, as it outperformed the C&R model in the first experiment.

An error analysis revealed that many of the errors of the Collins model in Experiment 1 are chunking errors. For example, the PP *neben den Mitteln des Theaters* should be analyzed as follows:



(3.4)

But instead the parser produces two constituents:



The reason for this problem is that *neben* is the head of the constituent in (3.4), and the Collins model uses a crude distance measure together with head-head dependencies to decide if additional constituents should be added to the PP. The distance measure is inadequate for finding PPs with high precision.

The chunking problem is more widespread than PPs. The error analysis shows that other constituents, including Ss and VPs often have incorrect boundaries. This problem is compounded by the fact that the rules in NEGRA are substantially flatter than the rules in the Penn Treebank, for which the Collins model was developed. Table 3.3 compares the average number of daughters in both corpora.

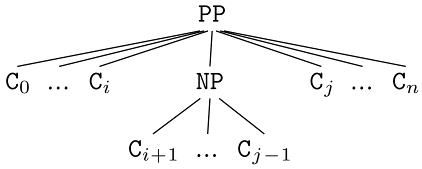
The flatness of PPs is easy to reduce. As detailed in Section 2.4, PPs lack an intermediate NP projection, which can be inserted straightforwardly using the following algorithm:

```
for a tree node that corresponds to the rule PP → C_0 ... C_n
let i = position of the last preposition, or -1 if there is no preposition
let j = position of the first postposition, or n if there is postposition
if j-i=0 or if j-i=1 and the i+1st constituent is a CNP,
    return the rule unchanged
```

	C&R	Collins	Charniak	Sister-Head
Head sister category	X	X	X	
Head sister head word	X	X	X	
Head sister head tag		X	X	
Previous sister category	X		X	X
Previous sister head word				X
Previous sister head tag				X

Table 3.4. Linguistic features in the sister-head model compared to the models of Carroll and Rooth (1998), Collins (1997) and Charniak (2000)

```
else return
```



In the first experiment of this section, we investigate if parsing performance improves if we test and train on a version of NEGRA on which the transformation shown in Figure 3.4 has been applied. In a second series of experiments, we investigated a more general way of dealing with the flatness of NEGRA, based on the Collins (1997) model for non-recursive NPs in the Penn Treebank (which are also flat). For non-recursive NPs, Collins (1997) does not use the probability function in (3.2), which conditions upon the head word of the head daughter. Rather, it uses the following derivation, which conditions upon the head word of the previous sisters:

$$
\begin{aligned}
P_{lex}(\text{RHS}|\text{LHS}) &= P\left(\boxed{\begin{matrix}L_m\\ t_{L_m}\\ w_{L_m}\end{matrix}}\ldots\boxed{\begin{matrix}L_1\\ t_{L_1}\\ w_{L_1}\end{matrix}}\boxed{\begin{matrix}H\\ \\ \end{matrix}}\boxed{\begin{matrix}R_1\\ t_{R_1}\\ w_{R_1}\end{matrix}}\ldots\boxed{\begin{matrix}R_n\\ t_{R_n}\\ w_{R_n}\end{matrix}}\middle|\boxed{\begin{matrix}P\\ t_P\\ w_P\end{matrix}}\right)\\
&= P\left(\boxed{\begin{matrix}H\\ \\ \end{matrix}}\middle|\boxed{\begin{matrix}P\\ t_P\\ w_P\end{matrix}}\right) \qquad (3.5)\\
&\quad\cdot\left[\prod_{i=0}^{m} P\left(\boxed{\begin{matrix}L_i\\ t_{L_i}\\ w_{L_i}\end{matrix}}\middle|\boxed{\begin{matrix}L_{i-1}\\ t_{L_{i-1}}\\ w_{l_{i-1}}\end{matrix}}\boxed{\begin{matrix}P\\ \\ \end{matrix}}\right)\right]\\
&\quad\cdot\left[\prod_{i=0}^{n} P\left(\boxed{\begin{matrix}R_i\\ t_{R_i}\\ w_{R_i}\end{matrix}}\middle|\boxed{\begin{matrix}R_{i-1}\\ t_{R_{i-1}}\\ w_{R_{i-1}}\end{matrix}}\boxed{\begin{matrix}P\\ \\ \end{matrix}}\right)\right]
\end{aligned}
$$

Using such *sister-head* relationships is a way of counteracting the flatness of the grammar productions; it implicitly adds binary branching to the grammar. Our proposal is to extend the use of sister-head relationship from non-recursive NPs (as proposed by Collins) to all categories.

Table 3.4 shows the linguistic features of the resulting model compared to the models of Carroll and Rooth (1998), Collins (1997), and Charniak (2000). The C&R model effectively includes category information about *all* previous sisters, as it uses context-free rules. The Collins (1997) model does not use context-free rules, but generates the next category using zeroth order Markov chains (see Section 4.2.3), hence no information about the previous sisters is included. Charniak (2000) model extends this to higher order Markov chains (first to third order), and therefore includes category information about previous sisters.The current model differs from all these proposals: it does not use any information about the head sister, but instead includes the category, head word, and head tag of the previous sister, effectively treating it as the head.

	Precision	Recall	F-score	Avg CB	0CB	$\leqslant 2$ CB	Coverage
Unmodified Collins	67.9	66.1	67.0	0.73	65.7	89.5	95.2
Split PP	73.8	73.8	73.8	0.82	62.9	89.0	95.1
Collapsed PP	66.5	66.1	66.3	0.89	66.6	87.0	95.1
Sister-head NP	67.8	66.0	66.9	0.75	65.9	89.0	95.1
Sister-head PP	70.3	68.5	69.3	0.69	66.3	90.3	94.8
Sister-head all	71.3	70.9	71.1	0.61	69.5	91.7	95.9

Table 3.5. Sister-head model with TnT tags

3.3.1 Method

We first trained the original Collins model on a modified versions of the training test from Experiment 1 in which the PPs were split by applying the rule from Figure 3.4.

In a second series of experiments, we tested a range of models that use sister-head dependencies instead of head-head dependencies for different categories. We first added sister-head dependencies for NPs (following the proposal of Collins, 1999) and then for PPs, which are flat in NEGRA, and thus similar in structure to NPs (see Section 2.4). Then we tested a model in which sister-head relationships are applied to all categories.

In a third series of experiments, we trained models that use sister-head relationships everywhere except for one category. This makes it possible to determine which sister-head dependencies are crucial for improving performance of the model.

3.3.2 Results

The results of the PP experiment are listed in Table 3.5 for TnT tags and Table 3.6 for perfect tags. The row 'Split PP' contains the performance figures obtained by including split PPs in both the training and in the testing set. This leads to a substantial increase in F-score (around 7%) for both tagging schemes. Note, however, that these figures are not directly comparable to the performance of the unmodified Collins model: it is possible that the additional brackets artificially inflate the F-score. Presumably, the brackets for split PPs are easy to detect, as they are always adjacent to a preposition. An honest evaluation should therefore train on the modified training set (with split PPs), but collapse the split categories for testing, i.e., test on the unmodified test set. The results for this evaluation are listed in rows 'Collapsed PP'. Now there is no increase in performance compared to the unmodified Collins model; rather, a slight drop in F-score is observed.

Tables 3.5 and Table 3.6 also display the results of the experiments with the sister-head model, with TnT and perfect tags, respectively. For TnT tags, we observe that using sister-head dependencies for NPs leads to a small decrease in performance compared to the unmodified Collins model, resulting in an F-score of 66.9. Sister-head dependencies for PPs, however, increase performance substantially to 69.3. The highest improvement is observed if head-sister dependencies are used for all categories; this results in an F-score of 71.1, which corresponds to an improvement of 4 points over the unmodified Collins model. Performance with perfect tags is around 2--4 points higher than with TnT tags. For perfect tags, sister-head dependencies lead to an improvement for NPs, PPs, and all categories.

	Precision	Recall	F-score	Avg CB	0CB	$\leqslant 2$ CB	Coverage
Unmodified Collins	68.6	66.9	67.8	0.71	65.0	89.7	96.2
Split PP	75.9	75.3	75.6	0.77	65.4	89.0	93.8
Collapsed PP	68.2	67.3	67.8	0.94	66.7	85.9	93.8
Sister-head NP	71.5	70.3	70.9	0.60	68.0	93.3	94.6
Sister-head PP	73.2	72.4	72.8	0.60	68.5	93.2	94.5
Sister-head all	73.9	74.2	74.1	0.54	72.3	93.5	95.2

Table 3.6. Sister-head model with perfect tags

	Perfect Tagging		TnT Tagging	
	ΔLR	ΔLP	ΔLR	ΔLP
PP	− 3.45	− 1.60	− 4.21	− 3.35
S	− 1.28	0.11	− 2.23	− 1.22
Coord	− 1.87	− 0.39	− 1.54	− 0.80
VP	− 0.72	0.18	− 0.58	− 0.30
AP	− 0.57	0.10	0.08	− 0.07
AVP	− 0.32	0.44	0.10	0.11
NP	0.06	0.78	− 0.15	0.02

Table 3.7. Change in performance when reverting to head-head statistics for individual categories

The third series of experiments was designed to determine which categories are crucial for achieving this performance gain. This was done by training models that use sister-head dependencies for all categories but one. Table 3.7 shows the change in LR and LP that was found for each individual category (again for TnT tags and perfect tags). The highest drop in performance (around 3 points) is observed when the PP category is reverted to head-head dependencies. For S and for the coordinated categories (CS, CNP, etc.), a drop in performance of around 1 points each is observed. A slight drop is observed also for VP (around 0.5 points). Only minimal fluctuations in performance are observed when the other categories are removed (AP, AVP, and NP): there is a small effect (around 0.5 points) if TnT tags are used, and almost no effect for perfect tags.

3.3.3 Discussion

We showed that splitting PPs to make NEGRA less flat does not improve parsing performance if testing is carried out on the collapsed categories. However, we observed that the F-score is artificially inflated if split PPs are used for testing. This finding goes some way towards explaining why the parsing performance reported for the Penn Treebank is substantially higher than the results for NEGRA: the Penn Treebank contains split PPs, which means that there are lot of brackets that are easy to get right. The resulting performance figures are not directly comparable to figures obtained on NEGRA, or other corpora with flat PPs.[3.1]

We also obtained a positive result: we demonstrated that a sister-head model outperforms the unlexicalized baseline model (unlike the C&R model and the Collins model in Experiment 1). F-score was about 4% higher than the baseline if lexical sister-head dependencies are used for all categories. This holds both for TnT tags and for perfect tags (compare Tables 3.5 and 3.6). We also found that using lexical sister-head dependencies for all categories leads to a larger improvement than using them only for NPs or PPs (see Table 3.7). This result was confirmed by a second series of experiments, where we reverted individual categories back to head-head dependencies, which triggered a decrease in performance for all categories, with the exception of NP, AP, and AVP (see Table 3.7).

3.1. This result generalizes to S's, which are also flat in NEGRA (see Section 2.4). We conducted an experiment in which we added an SBAR above the S. No increase in performance was obtained if the evaluation was carried using collapsed Ss.

On the whole, the results of this experiment are at odds with what is known about parsing for English. The progression in the probabilistic parsing literature has been to start with lexical head-head dependencies (Collins, 1997) and then add non-lexical sister information (Charniak, 2000), as illustrated in Table 3.4. Lexical sister-head dependencies have only been found useful in a limited way: in the original Collins model, they are used for non-recursive NPs.

Our results show, however, that for parsing German, lexical sister-head information is more important than lexical head-head information. Only a model that replaced lexical head-head with lexical sister-head dependencies was able to outperform a baseline model that uses no lexicalization.[3.2] Based on the error analysis for the first experiment, we claim that the reason for the success of the sister-head model is the fact that the rules in NEGRA are so flat; using a sister-head model is a way of binarizing the rules.

3.4 The Effect of Lexicalization

If it is indeed the case that flatter structures are causing the performance difference between the head-head and sister-head parsers, it is reasonable to ask if the difference is due to lexicalization at all. In both the head-head and sister-head version of the Collins model, lexicalization is closely tied to the 'structural model', i.e. the unlexicalized rule probabilities. Recall that the Collins model uses 0^{th} order Markovization to 'forget' previous sisters (cf. Section 3.1). The C&R model uses a similar trick, but only for lexical probabilities. Rule probabilities are left unchanged from the PCFG model.

Therefore, the effect of the lexicalization in the C&R model is easy to explain. Because we test the underlying PCFG separately, and because adding lexicalization lowers the performance, we can confidently say that it is lexicalization that is hurting the C&R model. We cannot make the same claim for the Collins model, because the underlying structural probabilities are so different from a PCFG. In this section, then, we test how well the unlexicalized backbones of the head-head and sister-head parsers perform alone.

3.4.1 Method

We use a similar experimental setup as in previous experiments. Given that we have found the difference in performance between TnT tagging and perfect tagging is predictable, we will only consider perfect tagging from this point onward. Of course, the models need to be modified to support unlexicalized parsing. The equations for unlexicalized parsing may be derived by removing the word features from the Collins and sister-head model. For the head-head model, we do this by changing Equation (3.2):

$$\begin{aligned}
P_{lex}(LHS|RHS) &= P\left(\boxed{\begin{matrix} L_m \\ t_{L_m} \end{matrix}} \ldots \boxed{\begin{matrix} L_1 \\ t_{L_1} \end{matrix}} \boxed{\begin{matrix} H \\ \end{matrix}} \boxed{\begin{matrix} R_1 \\ t_{R_1} \end{matrix}} \ldots \boxed{\begin{matrix} R_n \\ t_{R_n} \end{matrix}} \middle| \boxed{\begin{matrix} P \\ t_P \end{matrix}} \right) \\
&= P\left(\boxed{\begin{matrix} H \\ \end{matrix}} \middle| \boxed{\begin{matrix} P \\ t_P \end{matrix}} \right) \\
&\quad \cdot \left[\prod_{i=0}^{m} P\left(\boxed{\begin{matrix} L_i \\ t_{L_i} \end{matrix}}, d(i) \middle| \boxed{\begin{matrix} P \\ t_P \end{matrix}} \boxed{\begin{matrix} H \\ \end{matrix}} \right) \right] \\
&\quad \cdot \left[\prod_{i=0}^{n} P\left(\boxed{\begin{matrix} R_i \\ t_{R_i} \end{matrix}}, d(i) \middle| \boxed{\begin{matrix} P \\ t_P \end{matrix}} \boxed{\begin{matrix} H \\ \end{matrix}} \right) \right]
\end{aligned}$$

3.2. It is unclear what effect *bi*-lexical statistics have on the sister-head model; while shows bi-lexical statistics are sparse for some grammars, found they play a greater role in binarized grammars.

	Precision	Recall	F-score	Avg CB	0CB	$\leqslant$2CB	Cov
Head-head	68.45	67.32	67.9	0.60	66.98	92.91	96.21
Sister-head	72.38	69.72	71.0	0.61	65.90	93.49	97.05

Table 3.8. Results with lexicalization disabled (with perfect tags)

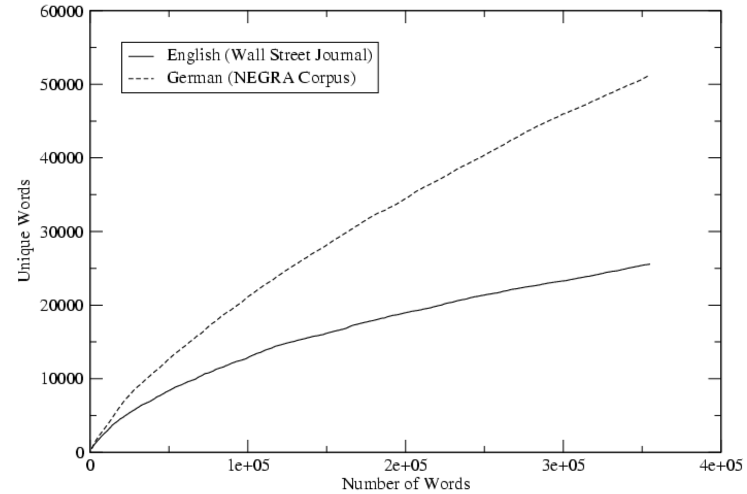


Figure 3.2. Unique words vs. number of words in NEGRA and the WSJ

Similarly, the new sister-head model becomes:

$$
\begin{aligned}
P_{sister}(LHS|RHS) &= P\left(\boxed{\begin{matrix} H \\ \end{matrix}} \middle\| \boxed{\begin{matrix} P \\ t_P \end{matrix}}\right) \\
&\cdot \left[\prod_{i=0}^{m} P\left(\boxed{\begin{matrix} L_i \\ t_{L_i} \end{matrix}} \middle\| \boxed{\begin{matrix} L_{i-1} \\ t_{L_{i-1}} \end{matrix}} \boxed{\begin{matrix} P \\ \end{matrix}}\right)\right] \\
&\cdot \left[\prod_{i=0}^{n} P\left(\boxed{\begin{matrix} R_i \\ t_{R_i} \end{matrix}} \middle\| \boxed{\begin{matrix} R_{i-1} \\ t_{R_{i-1}} \end{matrix}} \boxed{\begin{matrix} P \\ \end{matrix}}\right)\right]
\end{aligned}
$$

3.4.2 Results

The results of this experiment are shown in Table 3.8. The unlexicalized head-head parser achieves an F-score of 67.9. The score of the sister-head parser is slightly higher, at 71.0. By way of comparision, recall from Section 3.3.2 that the lexicalized version of the head-head and sister-head model acheive F-scores of 67.0 and 74.1, respectively. Coverage of both models was higher than many of the other models we have studied in this chapter.

3.4.3 Discussion

Curiously, the the unlexicalized head-head model outperformed the lexicalized version of the same model. In other words, lexicalization has almost no effect on the performance of the head-head model. Thus, we may reject sparse data as an explanation of the poor performance of the head-head model.

	English		German
I	sleep	ich	schlafe
you	sleep	du	schläfst
he	sleeps	er	schläft
we	sleep	wir	schlafen
you	sleep	ihr	schlaft
they	sleep	sie	schlafen
Total	2		4

Table 3.9. Number of word forms in present tense of "to sleep" in English and German

	English	**German**
Nominative	The *children* are here	Die *Kinder* sind hier
Dative	I talk with the *children*	Ich rede mit den *Kindern*
Total	1	2
Nominative masculine	A *young* man cheated Conrad	Ein *junger* Mann betrog Konrad
Accusative masculine	Conrad cheated a *young* man	Konrad betrog einen *jungen* Mann
Accusative feminine	Conrad cheated a *young* woman	Konrad betrog eine *junge* Frau
Total	1	3

Table 3.10. Number of word forms for example nouns and adjective in English and German

Once again, we appeal to the average branching factors listed in Table 3.3, and to the nature of the head-head model. Because the average branching factor is close to 2 in the WSJ, non-head dependants are usually adjacent to the head constituent. In NEGRA, this is typically not the case. With dependency-style grammars, recent constituents matter more than the head constituent: *recency matters*.

Turning our attention to the sister-head model, we find that the unlexicalized model tested here performs worse than the lexicalized model from Section 3.3. In other words, lexicalization helps. What is interesting, though, is that lexicalization does not help much. The difference between the F-score of the two models is only 3.1. This contrasts to results from English parsers, where lexicalization has been shown to significantly improve performance (cf. Collins, 1999). The result that lexicalization helps very little in NEGRA parsing has been replicated using a different model by .

We contend that part of the difficulty is due sparse data in the lexicalized grammar. Further, we argue the greater sparse data problem is caused by the larger number of word forms in German. Examples of this problem include verb conjugation (compare the number of unique conjugated forms of *schlaffen* 'to sleep' in the present tense in Table 3.9) and and noun declension (compare the number of forms of *Kind* 'child' and *jung* 'young' in Table 3.10). These specific examples are corroborated by plotting the type/token ratio of words in the NEGRA corpus in Figure 3.2.

3.5 The Effect of Flat Annotation

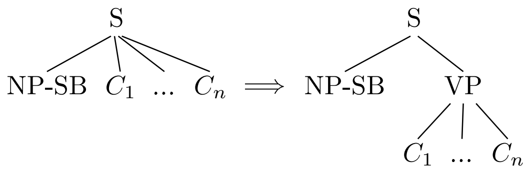
We have argued that sister-head parsing is more useful than head-head parsing because of the NEGRA annotation style. It would be insightful to test how well the parser works with less flat annotations, but on the same data. This would also provide is with clues to another question: why is the increase in performance due to lexicalization much greater in WSJ parsing than we have found in NEGRA parsing?

Strictly speaking, it would not be possible to do this without manually re-annotating the corpus. We can, however, approximate a less flat annotation scheme by semi-automatically modifying the corpus so that the annotations more closely resemble those of the WSJ. Doing this allows us to test if it really is the annotation style that causes the difference between the head-head and sister-head parser.

This re-annotation has another purpose, as well. Given that we use the same evaluation metrics that have become common in WSJ parsing, it is natural to want to compare our results on NEGRA with known results on the WSJ. But are the numbers really comparable? From our attempt to unflatten PPs in Section 3.3, we have some evidence this is not the case. But how great is the impact of dependency-like annotation on the evaluation metrics? In this section, we investigate this question, as well as the effect of dependency-style annotation on the two lexicalization strategies.

3.5.1 Method

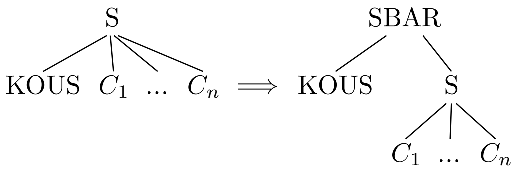
We have already investigated the impact of using some WSJ-style annotation, i.e. by applying Rule (3.4) to unflatten PPs. We propose three additional tree transformations, all affecting S categories. First, we introduce a VP category bounding the finite verb:



(3.6)

Although the parser we use in this section does not handle traces, one ought be inserted when the subject does not occupy position I (see Section 4.4 for more about topicalization).

The second transformation is to add an SBAR layer in complementizer phrases:



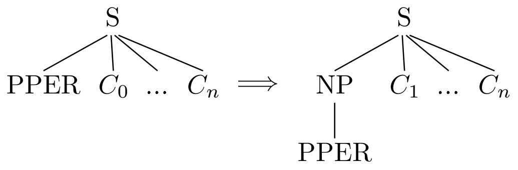
(3.7)

We treat subordinating co-ordinators (KOUS) and relative pronouns (PRELS, PRELAT and PWAV) as complementizers. Normally, the presence of a complementizer is both a sufficient and necessary condition for an SBAR, but there is one exception: co-ordination. For example, consider the sentence:

Wir	*reden*	[$_{\text{SBAR}}$	*weil*	*Ich*	*dumm*	*bin*	]	*und*	[$_{\text{SBAR}}$	*du*	*verrückt*	*bist*]
We	talk	[	because	I	stupid	am	]	and	[	you	crazy	are]

The complementizer is an empty element in the second SBAR. It can only be detected because it is a co-ordinate sister of the first one.

The third and final change involves pronouns and nouns. NEGRA does not contain unary productions, so any pronouns and singleton nouns will attach directly to an S node (or VP node) without the benefit of an intermediary NP node. We re-introduce these nodes. An example of the last transformation would be:



(3.8)

Note, though, that in addition to PPER tags, we also invoke this operation when any pronoun or stand-alone noun tag are found.

Each of the tree transformations aboves, along with the PP transformation from Section 3.3 are applied one at a time to the sister-head parser in the perfect tags condition.

	Precision	Recall	F-score	Avg CB	0CB	$\leqslant$2BC	Cov
Baseline	73.8	74.4	74.1	0.65	65.2	92.6	94.4
Split PP	76.4	76.7	76.5	0.88	60.2	88.0	93.4
Split VP	72.6	71.0	71.8	0.89	63.1	87.1	93.0
Split SBAR	74.0	75.0	74.5	0.70	65.4	91.1	94.1
Unary NP	76.0	76.4	76.2	0.64	65.6	92.7	94.3
PP+NP+SBAR	77.7	77.8	77.8	0.94	60.2	86.8	93.4

Table 3.11. Scoring effects on the sister-head model (with perfect tags)

	Precision	Recall	F-score	Avg CB	0CB	$\leqslant$2BC	Cov
Baseline	68.6	66.9	67.8	0.71	65.0	89.7	96.2
PP+NP+SBAR	77.7	77.8	77.7	1.03	58.5	85.1	93.4

Table 3.12. Scoring effects on the Collins model (with perfect tags)

Based upon the performance of these changes on the development set, we also apply the combination of three of the four transformations together. We leave out the VP transformation of Rule (3.6) in this case. This entails performing five experiments: each of the four transformations alone plus one experiment with three of the transformations together. We perform these five experiments on the sister-head parser. For the sake of comparison, we also perform the last experiment on the head-head parser.

3.5.2 Results

The results for the sister-head model summarized in Table 3.11. The first line shows the results of the baseline model, the sister-head parser without any modification. ‘Split PP’ refers to adding an NP node inside a PP, and this change raises the F-score to 76.5 from 74.1 for the baseline. The ‘Split VP’ line shows the result of adding a VP node dominating finite verbs. This transformation causes a dramatic fall in performance, to 71.8. The ‘Split SBAR’ operation provides a moderate improvement of 0.4 over the baseline. The last individual change, adding ‘Unary NP’ nodes improved performance to 76.2.

The combination of PP splitting, adding unary NP nodes and SBAR splitting improved performance of the sister-head parser to 77.8. The effect of the combined change on the Collins model is shown in Table 3.12. The average number of crossing brackets in the last condition is 1.03 with 58.5% of sentences having no crossing brackets and 85.1% of sentences having no more than two.

3.5.3 Discussion

Most of the unflattening operations helped. Moreover, the difference in performance of the sister-head and head-head model fell dramatically, from a difference of 8 points to a difference of about 2 points. This justifies the argument that much of the difference in performance between the head-head and sister-head parser in NEGRA are indeed due to assumptions about annotation. In addition, the higher overall scores appear to justify that scoring effects account for some of the differences between scores of the parser in NEGRA and the WSJ.

One interesting finding is that adding an explicit VP node dominating the finite verb did not help improve overall scores. It has been argued that freer-word order languages have an intrinsically flatter structure, and, in particular, that there is evidence that VP nodes do not exist in German . We are agnostic to this theoretical linguistic claim, but it is nonetheless interesting to point out that of all the unflattening operations we attempt, only this one actually hurt performance.

3.6 Verb Final Clauses

To this point, we have investigated how parsing is affected by the scoring effects, properties of the treebank and general properties of words. We have not looked into how the models cope with special syntactic properties of German. In this section, we do exactly that for one common construction found in German -- the verb final clause.

We briefly encountered this construction, which occurs in subordinate clauses, in both Section 1.1.1 as well as the preceding section. But let us examine it in more detail. In a subordinate or relative clause, the verb moves from its normal position in the second spot of the S rule to the last position in the rule, after any objects. Modifying the example from above, the main clause *Ich bin dumm* "I am stupid" becomes the subordinate clause *weil ich dumm bin* "because I stupid am".

Once again, because the sister-head model is the best performing model of those explored here, we will focus on that model here. In Section 4.4, we will evaluate the performance of various unlexicalized models with GFs on verb final clauses as well as sentences with subject movement.

This experiment is important because adding language-specific information is an important part of writing formal grammars, whereas the situation for treebanks-derived grammars might be different, if suitably advanced statistical parsing models can discern the distributions for verb-second and verb-final constructions on its own. We proceed with the hypothesis that the sister-head parser does indeed use such a model, and therefore verb-final clauses are no harder to parse than verb-second clauses.

3.6.1 Method

Our approach to evaluation is to split the test corpus into two parts, those sentences that have a verb-final construction and those which do not. We test the performance of the parser on each part, and use the difference as a metric to judge the relative ease or difficulty the parser encounters with such constructions. The intuition behind this split is that if a verb-final clause is harder to parse, the score of the whole sentence will be lower.

Dividing the evaluation corpus into the two required two parts is straightforward: verb-final constructions can be detected by the presence of a complementizer, as we saw in Section 3.5. In the development set, about 23% of sentences matched this criteria. Thus, one of the two parts is about three times larger than the other. Sentences with a verb-final clause also tended to be longer, and hence more complicated than other sentences. Thus, it may not be reasonable to make a direct comparison between the two parts.

Nevertheless, for the sake of exposition, the first evaluation metric we report is simply the F-scores of each part. The second metric, however, does attempt to take the complexity of a sentence into account, by applying a procedure to re-weight the scores of sentences.

The procedure works by keeping track of three sets of sentences: those sentences that contain a verb-final clause, those that do not, and an overall set that contains all the sentences from the first two. Let us refer to these cases as VF, NOVF and ALL. These sets are further subdivided into 'buckets' based upon the number of nonterminal nodes in a tree. For example, if a tree has 7 nonterminal nodes and it contain a verb-final construction, then it will go into bucket 7 of the vf and all cases.

The metrics we keep with this procedure are the components used to calculate the F-score: the number of correct nodes (call it C), the number of nodes in the gold tree (G) and the number of nodes in the proposed tree (P). We use subscript notation to denote the individual sums in each bucket. For instance, the total number of correct nodes in the verb-final case, in trees which have 7 nonterminal gold nodes would be represented by $C_{7,\mathrm{vf}}$.

	ALL	VF	NOVF
Average sentence length	7.5	11.4	6.6
Standard F-score	74.0	69.2	76.6
Weighted F-score	74.0	71.1	74.3

Table 3.13. Results on sentences with a verb-final clause with the sister-head model

To calculate the normal F-score, we would sum each of the three components C, G and P over all the buckets, then apply the usual precision, recall and F-score formulae. In this case, however, we give each bucket a weight before doing the summation. We calculate the weight in two steps. First, we normalize the C, G and P values for each bucket. This does not change the F-score within a bucket, but it does make the values comparable across buckets. Second, we ensure that each bucket contributes as much to the total as the matching bucket in the gold standard all case.

For example, in bucket 7 of the vf case, the normalization step entails dividing $C_{7,\mathrm{vf}}$ by $\sum_i G_{i,\mathrm{vf}}$, and likewise for $G_{7,\mathrm{vf}}$ and $P_{7,\mathrm{vf}}$. The second step involves multiplying the resulting number by:

$$w_{7,\mathrm{vf}} = \frac{G_{7,\mathrm{all}}}{\sum_i G_{i,\mathrm{all}}}$$

Continuing with the $C_{7,\mathrm{vf}}$ example, the 'weighted' version of this number is:

$$C'_{7,\mathrm{vf}} = C_{7,\mathrm{vf}} \cdot \frac{1}{G_{7,\mathrm{vf}}} \cdot \frac{G_{7,\mathrm{all}}}{\sum_i G_{i,\mathrm{all}}}$$

Using this weighting has two important properties. First, the F-score for the 'all' case does not change. Second, the F-score for each bucket in the other cases also remain invariant. The only change is the weighting that each bucket contributes to the overall sum.

While we have no guarantee that this approach to re-weighting will fully overcome the problem of different sentence lengths, we nonetheless report it to show how much this problem might be affecting the results.

3.6.2 Results

The results are summarized in Table 3.13. In the first line, we show the average length of a sentence in each set for illustrative purposes. Note, however, that the weighting scheme acts upon the number of nodes and not the number of words in a sentence. The F-score of sentences that contain at least one verb final construction is 69.2, compared to 76.6 to those sentences which contain no verb final construction, and 74.0 for all sentences together. Using our weighting scheme, the F-score in the verb final case rises to 71.1 and the score for sentences with no verb final clause falls to 74.3.

3.6.3 Discussion

The hypothesis that verb-final clauses are no harder to parse than verb-second clauses does not appear to have been vindicated by the data. Sentences with verb-final constructions appear to be much harder to parse than those without, even after re-weighting. It is important to remember, however, that the re-weighting scheme is provided primarily for illustrative purposes and is not a definitive method for accounting for the differences between the verb-final and non-verb-final parts of the test set.

Overall, this suggests that accuracy might be higher if the grammar includes some way of dealing with verb final clauses. We propose one such technique for unlexicalized parsing in Section 4.2, and evaluate it in detail in Section 4.4.

3.7 Conclusion

We have shown that lexicalized parsing models developed for English generalize poorly to NEGRA. Furthermore, we introduced a new model for parsing NEGRA based on sister-head relationships. These relationships emphasize recent sisters over the head sister, and we have shown sister-head dependencies are better suited for the flat structures in NEGRA. It also appears that the importance of recency over headedness is apparent even at the level of unlexicalized rules -- the sister-head model also outperforms the head-head model with lexicalization turned off.

The success of the sister-head model shows that lexicalization can be of benefit to German statistical parsers. It is worth comparing this result to results of similar models in other languages. As seen above, the sister-head model is a variant of the Collins models. Recall from Section 2.3 that the Collins model has been tested in several other languages besides English, including Czech (Collins et al., 1999) and Chinese (Bikel and Chiang, 2000). As we saw from Section 2.3, the performance attained by the model in these languages is lower than the performance in English. However, neither Collins et al. nor Bikel and Chiang compare the lexicalized model to an unlexicalized baseline model, leaving open the possibility that lexicalization is useful for parsing English text with Penn syntactic markup, but not for other languages or other annotation styles. In this chapter, we have explicitly tested this hypothesis, showing that lexicalization does indeed improve parsing accuracy, but not to the same degree as found in English. We explain the difference in degree by graphing word type/token ratio for NEGRA and the WSJ, which suggests that sparse data appears to be a bigger concern for German corpora than for English ones.

Even with lexicalization, overall results in German are lower than those in English. We found that some of this difference is due to scoring effects caused by annotation differences. Transforming the NEGRA treebank to look more like the Penn Treebank improves performance by about 8%.

Chapter 4

Grammatical Functions

In Chapter 3, we found that the parser which made use of grammatical functions (GFs) had a higher accuracy and lower coverage than other models. Unfortunately, the coverage was too low for this parser to be truly comparable to the other models. Nonetheless, the high accuracy of the GF parser does raise a number of interesting questions. For instance, what would happen if we could increase the coverage? And, more importantly, given the finding that lexicalization does not provide a big boost in performance (Section 3.4), could GFs offer comparable, or better performance than lexicalization?

Recall that the main objective of this thesis is to show that as morphological information becomes richer, the benefit of lexicalization becomes smaller, and that this deficiency can be overcome by including more linguistically-inspired features. Keeping with this overall objective, the goal of this chapter is to study the effect of grammatical functions as one possible feature to replace or augment lexicalization.

One could argue that including grammatical functions is a corpus-specific 'trick'. However, GFs play an important role in many linguistic theories, such as GPSG and dependency grammar. The use of GFs in dependency grammar is particularily interesting, as many treebanks (cf. the Prague and Dutch treebanks) are essentially dependency treebanks. While it is true that the Penn Treebank does not include nearly as many GFs as NEGRA, the GF labels in NEGRA are nonetheless relatively theory-neutral. By way of example, the three most common GF labels in NEGRA are subject, accusative object and modifier.

This chapter is organized as follows. In Section 4.1, we discuss a number of parsing models than can, unlike the model in the previous chapter, parse with grammatical functions and yet still maintain high coverage. Three main strategies are investigated: (i) giving the parser lexical sensitivity (but without full lexicalization), by integrating a POS tagger into the parser, (ii) using some generalization techniques such as Markovization and LP/ID rules to overcome missing rules and (iii) using a morphological analyzer to overcome tagging errors. We show not only that it is possible to develop a broad-coverage grammatical function parser, but also that some techniques that have been developed to parse the WSJ are annotation- and corpus-dependant. We also find that parsing with GFs is less accurate than parsing without them, at least with a simple model.

We provide a closer look at why this is so in Section 4.2. We detail a number of semi-automatic reannotations to grammatical functions which improve the grammar by including linguistic information that is not carried in raw grammatical functions. With this extra information, we develop a new parser with grammatical functions that can outperform a baseline parser which removes them. Moreover, this new parser is also more accurate on our test data than the "realistic" TnT-tagging sister-head model from Chapter 3. We also present evidence showing how the use of grammatical functions includes some of the same information gained by lexicalization.

Section 4.3 builds upon the parsers from Section 4.2, showing how sparse data problems can be overcome using smoothing. This is the first attempt we know of to use smoothing in an unlexicalized grammar. Using smoothing with unlexicalized grammar results in some unique problems, some of which we resolve, some of which has ramifications for other work. Rather than evaluating just one smoothing technique, we use several. This allows us to conjecture which approaches to smoothing are useful for unlexicalized parsing, and why.

In Section 4.4, we perform a detailed study of how well the best GF parsing models are able to cope with some well-known and difficult constructions in German. Building upon the work in Section 3.6, we test verb-final constructions as well as sentences which contain main clause subject movement. The evidence suggests that while the GF parsers do quite well overall, they are still troubled by these special constructions. Finally, in Section 4.5 we offer some concluding remarks.

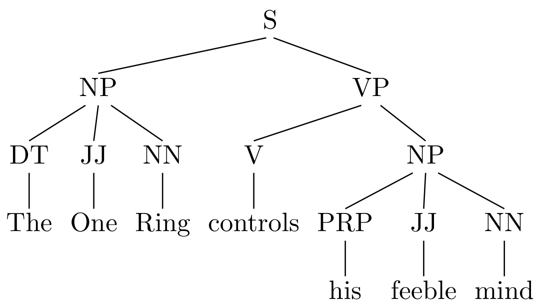
4.1 Parsing with Grammatical Functions

The GF parser investigated in Section 3.2 suffered from low coverage. It was unable to parse 35% of the sentences in the test set, and therefore it is not comparable to parsers which miss few or no sentences. In this section, we investigate two modifications which increase the coverage of that GF parser: (i) Markovization and (ii) integrating a POS tagger into the parser. As the second change requires that the parser uses words rather than POS tags as inputs, we may say that the change adds lexical sensitivity to the parser. Both of these modifications respectively lead itself to one additional change. Markovization can be generalized 'vertically', and the POS tagging may be improved by adding suffix analysis.

4.1.1 Markovization

The Collins and the sister-head models we saw in Chapter 3 both make use of *Markovization* to overcome the sparse data problems faced by lexicalized parsing models. However, Klein and Manning (2003) have shown that Markovization is also useful for unlexicalized models. It will be instructive to see if this result also holds in NEGRA. Johnson (1998) found that annotating a rule with its parent improves Penn Treebank parsing. Klein and Manning (2003) extend this idea by noting that, mathematically, the structure of PCFG rules seem arbitrary in that parents are treated differently than sisters. Thus, they proposed that in addition to *horizontal Markovization* (based on sisters), one may also wish to explore various kinds *vertical Markovization* (based on parents), noting that default context-free grammar rules already have the 1^{st} order vertical Markov property.

The best way to explain horizontal and vertical Markovization might be by way of history-based parsing Black et al. (2003). For example, consider the following tree:



Ignoring lexical probabilities, a history-based parser might assign probabilities to the tree in the following manner[4.1]:

$$\begin{aligned}
&P(\mathrm{S} \to \mathrm{NP} \ldots \mid \mathrm{S}) \quad (4.1)\\
\cdot\ &P(\mathrm{S} \to \ldots \mathrm{VP} \mid \mathrm{S} \to \mathrm{NP} \ldots)\\
\cdot\ &P(\mathrm{NP} \to \mathrm{DT} \ldots \mid \mathrm{S} \to \mathrm{NP}\,\mathrm{VP}, \mathrm{NP} \to \ldots)\\
\cdot\ &P(\mathrm{NP} \to \ldots \mathrm{JJ} \mid \mathrm{S} \to \mathrm{NP}\,\mathrm{VP}, \mathrm{NP} \to \mathrm{DT} \ldots)\\
\cdot\ &P(\mathrm{NP} \to \ldots \mathrm{NN} \mid \mathrm{S} \to \mathrm{NP}\,\mathrm{VP}, \mathrm{NP} \to \mathrm{DT}\,\mathrm{JJ} \ldots)\\
\cdot\ &P(\mathrm{VP} \to \mathrm{V} \ldots \mid \mathrm{S} \to \mathrm{NP}\,\mathrm{VP}, \mathrm{NP} \to \mathrm{DT}\,\mathrm{JJ}\,\mathrm{NN}, \mathrm{VP} \to \ldots)\\
\cdot\ &P(\mathrm{VP} \to \ldots \mathrm{NP} \mid \mathrm{S} \to \mathrm{NP}\,\mathrm{VP}, \mathrm{NP} \to \mathrm{DT}\,\mathrm{JJ}\,\mathrm{NN}, \mathrm{VP} \to \mathrm{V} \ldots)\\
\cdot\ &P(\mathrm{NP} \to \mathrm{PRP} \ldots \mid \mathrm{S} \to \mathrm{NP}\,\mathrm{VP}, \mathrm{NP} \to \mathrm{DT}\,\mathrm{JJ}\,\mathrm{NN}, \mathrm{VP} \to \mathrm{V}\,\mathrm{NP})\\
\cdot\ &P(\mathrm{NP} \to \ldots \mathrm{JJ} \mid \mathrm{S} \to \mathrm{NP}\,\mathrm{VP}, \mathrm{NP} \to \mathrm{DT}\,\mathrm{JJ}\,\mathrm{NN}, \mathrm{VP} \to \mathrm{V}\,\mathrm{NP}, \mathrm{NP} \to \mathrm{PRP} \ldots)\\
\cdot\ &P(\mathrm{NP} \to \ldots \mathrm{NN} \mid \mathrm{S} \to \mathrm{NP}\,\mathrm{VP}, \mathrm{NP} \to \mathrm{DT}\,\mathrm{JJ}\,\mathrm{NN}, \mathrm{VP} \to \mathrm{V}\,\mathrm{NP}, \mathrm{NP} \to \mathrm{PRP}\,\mathrm{JJ} \ldots)
\end{aligned}$$

This expansion would clearly overfit any kind of training data: every node is conditioned upon everything which occurred before it. To overcome this problem, we use the familiar technique of making independence assumptions. The most common such assumption, the one that results in PCFGs, posits that a node only depends on its parent and all its previous sisters.

Consider for a moment the following term from Equation 4.1:

$$P(\mathrm{NP} \to \ldots \mathrm{NN} \mid \mathrm{S} \to \mathrm{NP}\,\mathrm{VP}, \mathrm{NP} \to \mathrm{DT}\,\mathrm{JJ}\,\mathrm{NN}, \mathrm{VP} \to \mathrm{V}\,\mathrm{NP}, \mathrm{NP} \to \mathrm{PRP}\,\mathrm{JJ} \ldots)$$

Under the standard PCFG assumptions, we condition upon the immediate parent and all the previous sisters, resulting in the probability:

$$P(\mathrm{NP} \to \ldots \mathrm{NN} \mid \mathrm{NP} \to \mathrm{PRP}\,\mathrm{JJ} \ldots) \quad (4.2)$$

Note this implicitly makes the 1^{st} order vertical Markov assumption; if instead we made the 2^{nd} order vertical Markov assumption, we include two previous parents, resulting in:

$$P(\mathrm{NP} \to \ldots \mathrm{NN} \mid \mathrm{VP} \to \mathrm{NP}, \mathrm{NP} \to \mathrm{PRP}\,\mathrm{JJ} \ldots)$$

In other words, 2^{nd} order vertical Markovization is equivalent to doing a grandparent annotation as introduced by Johnson (1998)[4.2]. All these simplified rules still depend on all previous sisters, akin to making an infinite-order horizontal Markov assumption. If, on the other hand, we wish to make, say, a 1^{st} order horizontal Markov assumption from Equation 4.2, we would get:

$$P(\mathrm{NP} \to \ldots \mathrm{NN} \mid \mathrm{NP} \to \mathrm{JJ} \ldots)$$

The key idea behind Markovization is to "forget" far away information in flat rules. There are other, and perhaps linguistically more principled ways of accomplishing this. One approach is to use linear precedence/immediate dominance (LP/ID) rules Gazdar et al. (1985). Just as with Markovization, using LP/ID rules reduces the number of rules compared to a standard context-free grammar. LP/ID rules acheive this by "relaxing" the restrictions of context-free rules. A context-free grammar rule consists of a parent and an ordered list of children. In contrast, LP/ID rule consists of a parent and an unordered multiset of children (the immediate dominance part of the rule). In addition, a partial ordering can be specified by listing violatable constrains which specify which children may come before others (i.e. subject before object). This is the linear precedence part of the rule.

4.1. The ...'s are akin to the dot in a dotted rule. That is, they denote that the rule expansion is incomplete.

4.2. Johnson refers to it as 'parent annotation'; we find the term 'grandparent annotation' a bit more intuitive.

There are a number of ways to create a probability distribution of LP/ID rules. We use a simple approach, similar to that used in Model 2 of Collins (1999). Just like the probability distribution in a (horizontal) Markov grammar, nodes are added one at a time. However, the information a new node is conditioned upon is slightly different in two respects. First, we condition on all nodes. Contrast this with an nth order Markov grammar, which conditions upon the previous n nodes. Second, the order of the previous nodes is ignored. In other words, we condition on the *multiset* rather than a list of previous children. By conditioning on a multiset, we model the ID part of the rule; by adding children one at a time, we model the LP part.

4.1.2 Lexical Sensitivity

The unlexicalized parsers in Chapter 3 took POS tags as input, ignoring the actual words of the input sentence. Obviously, the lexicalized parsers did not ignore the words, but they also took a fixed set of POS tags as input. Neither approach is appropriate for the models we develop here, because the set of POS tags is considerably more expressive than in Chapter 3.

In the NEGRA corpus, POS tags are annotated with GF labels. Because grammatical functions express syntactic information which may be ambiguous even in the ± 2 word window used by POS taggers, it is not reasonable to assume that a POS tagger can accurately guess both the tag and the grammatical function. For example, a single word NP may contain a grammatical function indicating the case of the NP, but the case may be difficult for a finite-state model to guess without knowing where the NP occurs in relation to the main verb and its other arguments.

But this is not the only reason to integrate a POS tagger into the parser. Not only would it be hard for a tagger to apply such tags accurately, but it may actually *help* the parser if it were to apply tags directly. This is because allowing the parser to tag itself affords it a degree of lexical sensitivity (cf. Section 2.2.1).

4.1.3 Suffix analysis

Many finite-state POS taggers guess POS tags based on the prefix or suffix of previously unseen words. The terms 'prefix' and 'suffix' are used rather loosely, and mainly refer to looking at some letters from the end or start of a word. Even though this information is simple, it has proven to be useful for guessing the POS tags of rare or unseen words (Brill, 1995; Ratnaparkhi, 1996; Brants, 2000). As our parser will attempt to do POS tagging itself, it is reasonable to expect the parser's POS distribution benefits from the techniques used by finite-state POS taggers.

We use the suffix analyzer introduced by Brants (2000). This particular analyzer has three advantages over others: it uses a simple maximum likelihood distribution that integrates well with our parser, it was developed with German in mind, and finally, the POS tagger of Brants (2000) is one of the most accurate, leading us to suspect the suffix analyzer also works quite well.

Brants' suffix model works as follows: for a word w with letters $l_1, l_2, ..., l_n$ we compute the probability of a tag t using the last m letters. Recursively, we compute:

$$P(t|l_{n-i+1}...l_n) = \frac{P(t|l_{n-i+1}...l_n) + \theta_i P(t|l_{n-i}, ..., l_n)}{1+\theta_i}$$

Where

$$\theta_i = \frac{1}{s-1}\sum_{j=1}^{s} [P(t_j) - \bar{P}]^2$$

Bayes' Law is then used to find the generative probability $P(l_{n-m+1}...l_n|t)$. This probability is used as an approximation of $P(w|t)$.

4.1.4 Method

The experiments here use the same data sets as in Chapter 3. For these experiments, we use an exhaustive CYK parser. With this parser, we execute permutations of the alternatives listed above. We report the result of 24 of these permutations, resulting from choosing one possibility from each of the following four categories:

Horizontal assumption. Three possible choices: normal rules (∞ order horizontal Markovization); 2^{nd} order horizontal Markovization; LP/ID rules.

Vertical assumption. Two choices: either 1^{st} or 2^{nd} order vertical Markovization. In other words, either conditioning on just the parent like in a PCFG, or on the parent and the grandparent, as in Johnson (1998).

Suffix assumption. Two choices: either an unknown word distribution (as with the unlexicalized parser in Chapter 3) or with TnT-style suffix analysis.

GF assumption. Either with or without grammatical functions.

There are a number of experiments we do not report here, for example other possible values of n for n^{th} order horizontal and vertical Markovization. Preliminary experiments on the development set, however, have shown that these alternative settings are all less accurate, and therefore less interesting than what we present here.

4.1.5 Results

Perhaps the most important result concerns the coverage of the various models. Overall, coverage of the no-GF models were quite high, with between 0.3% and 3.3% of all parses missed. Coverage of the GF models was slightly lower, with between 0.9% and 2.5% of all parses missed. This is quite an improvment over the GF model of Chater 3, which missed 35% of all sentences. The increase in coverage is solely due to the decision to allow the parser to assign its own tags to the input: horizontal Markovization and LP/ID did not provide a large improvement in coverage compared to standard CFG rules.

The accuracy of the models are shown in Tables 4.1 and 4.2. Table 4.1 details the results without grammatical functions and Table 4.2 shows the results with grammatical functions. For the sake of simplicity, we only show the F-score of each parser. However, the crossing bracket measures tend to be correlated with the F-scores.

The tables are laid out similarly. Each column contains the result of a different horizontal rule assumption, and each row contains the result of a different vertical and suffix assumption. In these experiments, horizontal Markovization is more accurate than standard CFG rules or LP/ID rules, vertical Markovization does not appear to be useful when GFs are used, and adding a suffix analysis only helps without grammatical functions.

4.1.6 Discussion

The two primary goals of the experiments above were to (i) increase the coverage of, and to (ii) maintain the same accuracy as the perfect POS tag GF parser from Chapter 3. It appears, however, that we achieved the goal of coverage at the expense of the goal of accuracy. It may not be surprising that the accuracy fell, however. As noted earlier, NEGRA POS tags contain GF labels, so using perfect POS tags (as in Chapter 3) resolves some parsing ambiguities.

While it may not be surprising that the performance of the GF parsers are lower than the perfect tag GF parser of Chapter 3, it is surprising that most of the parsers which include GFs perform worse than their cousins without GFs. For instance, in the case with suffix analysis, conditioning on the parent, and with the 2^{nd} order Markov property, the parser with GFs scored 67.7, while the parser without GFs scored 70.7.

	∞ order Markov	2^{nd} order Markov	LP/ID
No suffix, parent	66.8	68.6	66.9
No suffix, grandparent	67.5	69.4	67.3
Suffix, parent	71.0	**72.5**	70.7
Suffix, grandparent	64.0	67.5	64.0

Table 4.1. Results (F-Scores) when GFs are excluded

	∞ order Markov	2^{nd} order Markov	LP/ID
No suffix, parent	66.3	**69.4**	66.5
No suffix, grandparent	65.2	68.7	65.8
Suffix, parent	66.2	69.1	66.6
Suffix, grandparent	61.2	64.4	61.6

Table 4.2. Results (F-Scores) when GFs are included

This reduction in performance when using GFs may be justified by the theoretical importance of GFs. In some respects, a model with GF tags is superior to a better-performing model without such tags because GFs help in resolving ambiguities caused by word-order flexibility. Indeed, the importance of GFs in sentence understanding compel us to evaluate the performance of GF labelling in Chapter 6.

But the usefulness of GFs for handling word-order flexibility makes it more surprising that GFs do not convey any useful parsing information at the level of syntactic categories. However, a more detailed examination of the precision and recall of individual categories (cf. the first two columns of Table 4.5) shows that the decline in performance when using GFs is not equal across categories. While there are substantial decreases in the common categories S, NP and PP, the biggest drop in performance is concentrated in co-ordinated categories. Also, we can see that GFs do not help the scores of NPs -- another surprising fact as one would expect that GFs (which encode some case information) would be beneficial in finding the boundaries of NPs. In Section 4.2, we investigate strategies of overcoming both these problems which, as we shall see, are both related to problems of *communication* between a parent and child node.

Grandparent annotation, originally devised as one way of increasing communication between parent and child nodes, was not helpful. This is at odds to what has been reported for the Penn Treebank: Johnson (1998) shows that grandparent annotation yields labelled bracketing results 5 percentage points higher than without. However, Johnson's analysis of his results probably point to one reason why his result fails to generalize to NEGRA. He notes that the benefit is concentrated in NP's and VP's. Subject NP's are more likely to expand to a unary node than object NP's and non-finite VP's behave quite differently than the VP of the finite verb. In both cases, we cannot tell the difference between a subject/object NP or a finite/non-finite VP using the basic Penn Treebank annotation, but we can make the difference if the grandparent is annotated: the grandparent is usually an S for a subject NP or finite VP and it is usually a VP for an object NP or non-finite VP.

While it is true that NEGRA usually does not contain a VP nodes, making it hard to make the equivalent distinction, it is also true that this is not an important ambiguity in NEGRA. Since unary nodes expand directly to the nonterminal, in the unary subject/non-unary object ambiguity in NP is missing altogether. In the finite/nonfinite case, we need not worry about there being two different kinds VP categories: finite verbs attach directly to the S constituent.

Thus, at least in the two major categories pointed out by Johnson, it appears as if grandparent node annotation is not necessary in NEGRA. We may expect the same to be true for other dependency-style treebanks for the finite/nonfinite distinction. However, this result may not be true for other dependency-style treebanks in the subject/object case. The decision to eliminate all unary nodes in NEGRA is strictly an annotation-depdendant choice. However, even if unary nodes had not been eliminated, grammatical functions may well have carried the same information as grandparent annotation.

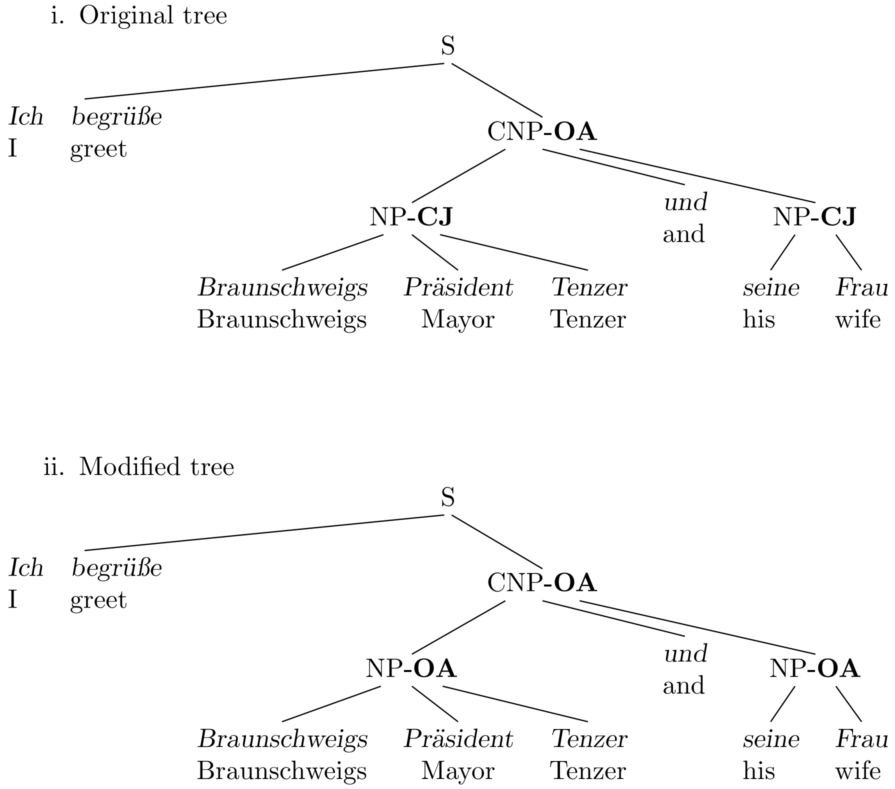


Figure 4.1. The co-ordination re-annotation operation

Turning our attention to suffix analysis, we find it helped considerably with the simpler grammars. But when grandparents or grammatical functions were added, doing a suffix analysis ended up decreasing performance. This might very well point to a sparse data problem. The data sparsity, however, might not be with the suffix analyzer itself, but with the grammar: the suffix analyzer may have a strong tendency for a certain tag, and although the tag is good, the grammar simply doesn't have a good enough rule to go with it.

With suffix analysis as well, then, we return the problem that grammatical functions do not seem to work very well, and we need some way to improve their performance.

4.2 Grammatical Function Re-annotation

The rather disappointing result of Section 4.1 may actually have to do with a number of choices of the NEGRA annotation scheme, particularly the choice of what syntactic phenomena should or should not be included in the annotation. To understand why this is true, we must analyze what information grammatical functions give to the parser, and how the parser can make the best use of this information.

Let us begin this analysis with an example. Consider the OA (accusative object) and OD (dative object) grammatical functions. Both apply to NPs, but they differ in the places where they can appear in a rule (accusative objects generally occur before dative objects) and in the strings they generate (*einen Mann* in the accusative versus *einem Mann* in the dative). However, the annotation in NEGRA only allows us to 'see' differences at the rule level.

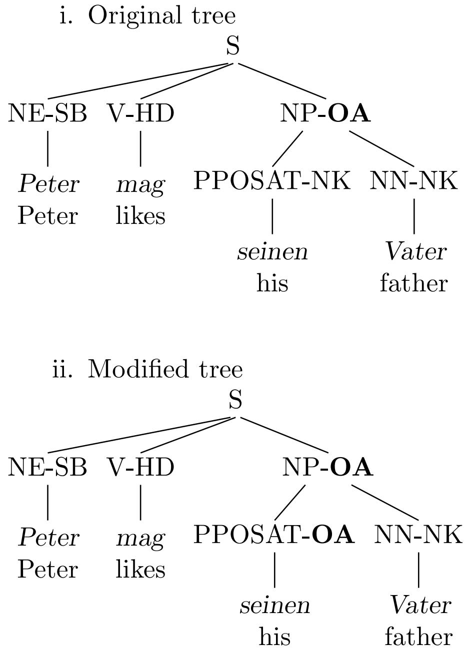


Figure 4.2. The NP re-annotation operation

In this case, the problem is with POS tags. While the OA and OD GF labels indicate that an NP is either accusative or dative, all POS tags dominated by NPs simply get an NK (noun kernel) GF, as shown in Tree i of Figure 4.2. This makes the implicit assumption that the rules “DT-NK → einen” and “DT-NK → einem” are equally likely to be seen in an accusative NP as they are in a dative NP because our grammar only contains rules like “NP-DA → DT-NK ...” and “NP-OA → DT-NK ...’ This problem can be easily solved by copying the function of the parent to the relevant children, as depicted in Figure 4.2, Tree ii. In addition to determiners, we apply the case marking transformation to all pronouns, including demonstrative, indefinite personal, possessive (shown in Figure 4.2) as well as relative pronouns. One might argue that NK is indeed the correct grammatial function of POS tags like DT, and the correct approach to modelling the difference between *einen* and *einem* is to use a separate layer of morphological tags. This is a compelling argument which, in fact, forms the basis of Chapter 5.

Articles and pronouns are not the only parts of speech which indicate case. In NPs, nouns and adjectives may do so as well, but we do not consider that case here. However, prepositions also determine the case of the NP they take as a complement. German prepositions can be split into classes based upon the case they require: (i) those that take the accusative, such as *für* (“for”) or *um* (“around”); (ii) those taking the dative, such as *von* (“of”) or *mit* (“with’)’; (iii) those taking the genitive, such as *innerhalb* (“within”); (iv) those ambiguous between the dative and accusative, such as *in* (“in”) or *aus* (“from”); (v) those ambiguous between the genitive and a dative, such as *wegen* (“because of”) or *trotz* (“in spite of”); (vi) those which do not govern case, such as *als* (“as/than”). In cases (i)-(iii) we can easily mark the necessary information in the grammatical function tags, as shown in Figure 4.3 (Notice we mark up the same POS tags as with the NP transformation above). Cases (iv) and (v) are harder. Our approach is to introduce new functional tags, AD for ambiguous between accusative and dative, and DG for ambiguous between dative and genitive. This way, we still make the necessary distinction without having to introduce new information that is not annotated in the corpus. Case (vi) is left unchanged, with the preposition still carrying the -AC tag and the determiners and pronouns in the PP still carrying the -NK tag.

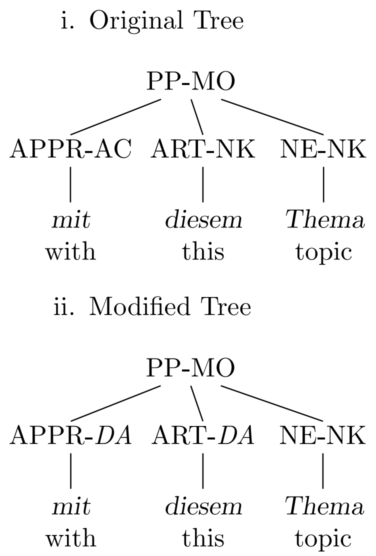


Figure 4.3. Grammatcal Functions and PP Case

We justified the co-ordination re-annotation operation because we noted a substantial drop in the performance of coordinate categories when adding grammatical functions, but it is important to realize that coordinate categories are not as frequent as their non-coordinated counterparts. A small change in performance in a common category can have a substantial change in overall results due to frequency weighting. Of the three most common categories, we have already proposed changes for NPs and PPs. But we have not yet explored the most common category of all -- the S category.

As we saw in Section 3.6, a prominent feature of German syntax is the word order of subordinate clauses. However, with some exceptions (such as the -RC function for relative clauses), this information is not carried in grammatical functions. While the function tags for Ss are no doubt important for semantic interpretation, they appear to carry little information that might be useful for detecting verb-final constructions.

Thus, in these cases, we replace the S label with an SBAR label. This is different from the SBAR reannotation of Section 3.5 in that we do not unflatten the tree; we simply rename the node. To account for co-ordination, we also rename CS to CSBAR, and rename any S children of CSBAR to SBAR. In Section 3.6, we called for a change to the grammar to account for verb-final constructions; this modification provides exactly such a change.

Because the GFs of the S categories may have little impact on the distribution of the children of the S, one final change we propose is to remove the grammatical functions of the S category altogether. For completeness, we also remove the GFs of CS, SBAR ans CSBAR categories.

Co-ordinated NPs have a similar problem. All co-ordinate sisters have the CJ grammatical function (cf. Tree i. of Figure 4.1). While technically true, this obscures more useful information: co-ordinate sisters must have the same case. To account for this, we simply replace all CJ tags with the tags of their parent, as shown in tree ii of Figure 4.1. This particular change can actually be applied to all categories, and not just NPs.

The experimental evidence from Section 4.1 suggests that the parsers using GF have a particularily difficult time with co-ordinated categories. Indeed, one large factor contributing to the difference in performance of the GF and no-GF models was the poor performance of co-ordinated categories.

Preposition case	Four most common GFs
Accusative	-MO (2594), -MNR (1880), (none, 71), -APP (23)
Dative	-MO (8619), -MNR (4189), -PG (977), -SBP (439)
Ambiguous	-MO (7055), -MNR (2923), (none, 149), -PD (99)

Table 4.3. The four most common grammatical functions for PPs, by case of the preposition

4.2.1 Method

For this experiment, our two baselines are the best-performing GF and no-GF grammars of the previous section: the 2$^{\text{nd}}$ order Markov PCFGs without grandparent annotation, but with morphological guessing. We test these two models against new grammars with the modifications illustrated above. The changes are tested incrementally, in order of presentation (results on the development set have shown the changes are, indeed, beneficial). In particular, we add to the baseline GF model coordinate copying, article function copying, PP case marking, S function deleting, and, finally, verb-final S marking.

4.2.2 Results

The overall results are summarized in Table 4.4, with a category-by-category listing in Table 4.5. The coordination and article copying operations improved the accuracy of the GF parser, resulting in a final *F*-score of 71.5 for the coordination copying and 72.4 for both together. Not only is the latter score an improvement of the baseline GF parser, which achieves and *F*-score of 69.1, but it also matches the no GF baseline. Moreover, all the crossing bracket measures were better than the no-GF parser: average CB is lower, at 0.70 versus 0.84; 66.2% of sentences had no crossing brackets versus 62.1%; and 90.9% of sentences had fewer than 2 crossing brackets, versus 88.5.

Annotating the case of PPs resulted in a small additional improvment. While marking SBARs slightly lowered the *F*-score for labelled bracketing, it greatly improved the average crossing bracket scores (it improved both scores on the development set). Removing the functional tags on the S categories gave another substantial boost, up to 73.0 in *F*-Score, along with small gains in the crossing bracket measures and in coverage. Note that this model also narrowly outperforms the sister-head model with TnT tags (although it still has a lower *F*-Score than the sister-head model with perfect tags).

The more detailed results in Table 4.5 show that coordinate copying is effective in removing the performance drop when grammatical functions are included. Much of the benefit from article tag GF copying is, as expected, in NPs, where there *F*-score rises from 61.0 without copying up to 64.2. Given that NPs are a high-frequency node, this difference has a substantial effect in the overall scores.

	Precision	Recall	*F*-Score	Avg CB	0 CB	$\leqslant 2$ CB	Coverage
No GF Baseline	72.0	72.9	72.5	0.72	65.2	90.7	99.8
GF Baseline	66.2	72.4	69.1	0.84	62.1	88.5	98.9
+Coordination Function	69.2	74.0	71.5	0.72	65.0	89.6	99.0
+NP Case Marking	69.9	75.1	72.4	0.70	66.2	90.9	99.1
+PP Case Marking	70.1	75.4	72.7	0.73	66.6	91.3	99.1
+SBAR Marking	70.0	75.3	72.6	0.67	66.5	92.1	98.9
+S Function Removing	70.8	75.5	**73.1**	0.67	66.4	91.9	99.1

Table 4.4. Labelled bracketing scores on the test set

	No GF	GF	Coord	NP
AP	46.6	48.8	49.8	48.4
AVP	21.0	29.3	31.8	32.1
MPN	74.9	79.8	79.8	79.0
NM	79.1	89.9	89.2	89.2
NP	64.9	62.9	65.0	66.5
PP	72.2	69.8	71.6	72.7
S	75.8	68.6	73.9	73.9
VP	54.1	52.6	53.9	55.0
CAP	62.6	42.1	68.5	63.0
CNP	62.1	44.1	62.3	59.9
CPP	43.4	45.4	40.0	40.0
CS	37.8	30.1	49.1	47.7
CVP	50.0	44.8	52.0	49.0

Table 4.5. Category-by-category listing

4.2.3 Discussion

While the improvement of each individual change was quite small, the overall improvement was much more substantial. One could argue the co-ordination copying function is annotation specific, however, most of the other reannotation operations are likely applicable to other depdency-style treebanks. We argue this is so because the operations put additional linguistic information into the grammar, usually dealing with case. There is one notable exception: the operation that deletes the grammatical function from S categories.

In some ways, it is problematic that this operation helps. It is not consistent to include this operation in a parser which claims to use grammatical functions: as S nodes are the third most common category, and after this operation they have no GF label at all.

To support the general claim that GFs are useful, it is important to examine why removing the GFs is a helpful modification. There are two possible reasons why this might be the case: (i) sparse data, and (ii) the possibility that GFs on the S node are simply not useful for the kind of syntactic disambiguation we are doing. In Section 4.3, we find some evidence that sparse data might be a factor: some smoothed models with GFs on S nodes do perform better than their counterparts without. But as this is not the case for all of the models, consider for a moment the possibility that GF labels are simply not useful here.

If this were the case, there would still be an argument for leaving the GF tags out, even if they might be useful for semantic interpretation: we could simply re-insert the labels after parsing. Indeed, this is the strategy used by Blaheta and Charniak (2000), henceforth B&C. B&C do not use the NEGRA corpus, but rather they work with the more sparsely annotated "functional labels" of the WSJ section of the Penn Treebank. Klein and Manning (2003) have shown that most of the grammatical functions in WSJ are not, in fact, useful for parsing. While the data are different than what we use, the results of B&C give credence to their approach: if the GF information is desired for semantic interpretation but is not useful for parsing, there may be better ways of getting it rather than putting it in the parser.

While GFs appear to be useful for PPs and NPs, there are also instances where the use of GFs might be harmful for these two categories. Consider NPs. We claim GFs help because they model case, and yet while there only are 4 cases in German, there are more than 10 possible GFs for an NP node. For instance, the -APP (apposition) tag is used for certain types of NP modifiers, regardless of their case. Thus, it tells us very little about the distribution of the case markers it dominates. This is part of a broader issue of syntactic function vs. syntactic distribution that we discuss in more detail in Chapter 5.

In contrast with NPs, prepositional phrases have a much weaker correlation between case of the NP it dominates and the GF of the parent category. The most common GF is -MO (modifier), and the second most common is -MNR (postnominal modifier), regardless of the case of the NP the preposition controls. However, a closer look at the GFs of various types of PPs tells a slightly different story. In Table 4.3, we list the three most common PP cases (if the ambiguous dative/accusative PPs are counted together). The table illustrates that the -PG (pseudo genitive) and -SBP (passive subject) GFs are strong predictors of PPs taking a dative NP. We would expect these annotations are correlated with a "light" lexicalization of annotating PPs with the head preposition. This is also true of determiner and pronoun GF copying in NPs. All in all, keeping better track of GF information is akin to lexicalizing many closed-class words, but requiring far less data.

4.3 Smoothing

While the unlexicalized parsers developed thus far are competitive with some of the lexicalized model we have considered, none is yet able to outperform the sister-head model from Chapter 4. In some ways, the comparison is not entirely fair. The sister-head model makes extensive use of smoothing, whereas the unlexicalized parser does not.

While smoothing in lexicalized models is justified on the grounds that these models have too many features to estimate reliably, the same might be said of GF parses we are investigating. In addition to the extra parameters due to the grammatical functions themselves, the sister-head parser only makes a 1^{st} order Markov assumption while the unlexicalized GF parsers do best with a 2^{nd} order assumption.

4.3.1 Search

Introducing smoothing into a parser necessitates changes to the parsing algorithm. In the worst case, many CFG parsing algorithms are $O(n^3)$. In practise, parsing is still quite fast because the grammar prunes out impossible parse derivations. As noted in Chapter 3, this is not true of smoothed grammars: every possible rule has some non-zero probability.

How many rules will we have? Using a grammar with the rth order Markov property (making all rules r nonterminals long) having m nonterminals, there may be upto m^{r+1} possible rules (m^r for the previous sister, and an extra m for the parent).

The number of rules impacts the number of edges the parser must visit. If there are n words in a sentence, then dynamic programming parsers will construct $O(n^2)$ edges per rule. If we pick $r = 2$, as above, then for the NEGRA GF grammar, where m is already over 256, we could not even fit the edges into a 32-bit address space for a sentence with just one word.

Clearly, an exhaustive search is not feasible. At least three possible alternatives to exhaustive searches have been suggested: greedy best-first parsing (Charniak and Caraballo, 1998), the $A^\star$ algorithm (Klein and Manning, 2002) and pruning (Goodman, 1998). Both best-first parsing and the $A^\star$ parsing use agenda-based chart parsers. The basic idea behind both approaches is to dictate the order that edges are pulled off the agenda. Pruning, on the other hand, can be used with any parsing algorithm. Indeed, this is the approach used in Chapter 3. At various stanges in the parsing algorithm, edges which are deemed unsuitable are discarded. As we have been using the CYK algorithm, the pruning approach lends itself well to be used with our implementation.

There is a downside to using pruning (or even best-first parsing) over the $A^\star$ algorithm: $A^\star$ is a *sound* algorithm, meaning that it is guaranteed to return the Viterbi parse. The results are the same as exhaustive parsing. Pruning and best-first parsing cannot guarantee this, and often do return parses with a lower probability than the Viterbi parse. It turns out, however, that pruning and best-first parsing are often more accurate than either exhaustive or $A^\star$ parsing.

Goodman (1998) discusses several types of pruning, which vary in complexity and in their success in making parsing faster. We test two approaches to pruning, beam search and multi-pass parsing. Both of these suitably trade-off simplicity with parsing speed.

Beam search Recall that the sister-head parser did use beam search. Like the sister-head parser, we use a prior probability together with the probability of a subtree to generate a figure-of-merit (FOM) for the subtree. There are two ways to implement a beam search. The first strategy, a *variable-width* beam, prunes any edges whose FOM falls below a certain threshold. The threshold is a multiple (say, $\frac{1}{4000}$) of the best FOM for that span. This is the same technique used in the sister-head parser. The second approach, a *fixed-width* beam, ranks edges according to their FOM and keeps only the n highest ranked edges per span. In preliminary tests, we found that a fixed-width beam is superior to a variable width beam for our grammars. This results differs from that reported by Brants and Crocker (2000). The difference may be because Brants and Crocker use a more compact representation for edges, meaning that what they consider to be one edge, we would consider to be several different edges. This would have a profound impact on a fixed-width beam.

Multipass parsing The key insight behind multipass parsing is that we can parse a sentence several times, using information from earlier parses to prune edges from later parsings. This strategy works because we use a grammar that are simpler (and hence faster) in the earlier passes. In our case, we use two-pass parsing with an unsmoothed grammar in the first pass.

4.3.2 Cached parsing

Pruning removes unwanted edges, but the memory requirements of CYK parsing still depend on the number of rules[4.3], not the number of edges. Following the definitions of Section 4.3.1 above, there are m^{r+1} rules, so the CYK algorithm still requires $O(n^2 \cdot m^{r+1})$ memory, even after implementing pruned searching.

This memory is required for the dynamic programming array (henceforth DPA), a three dimensional array indexed by (i) the starting word of an edge, (ii) the ending word of an edge and (iii) the rule number. The approach used by many lexicalized parsers (including Collins, 1999, and hence the sister-head parser of Chapter 3) is to use a hashtable to store the DPA. Due to pruning, the hashtable is normally quite sparse for lexicalized parsers, making parsing quite efficient.

In preliminary work with the GF parser, we found that the hashtable was not sparse enough to guarantee parsing efficiency. Storing the DPA in hashtables lead to parsing times in the hours rather than minutes. However, a closer observation of the algorithm lead to the insight that the array is accessed in two different ways in different parts of the algorithm. Recall the recursive part of the CYK algorithm:

```
for s=2 to n
  for i=1 to n−s−1
    let k=i+s
    for j=i to k-1
      for A=1 to #nonterms
        for B=1 to #nonterms
          for C=1 to #nonterms
            let p ← P(A→BC)·D[i,j,B]·D[j+1,k,C]
            if p > D[i,k,A] then
              D[i,k,A] ← p
```

4.3. In a binarized grammar, it actually depends on the number of nonterminals.

```
11:                     backtrace[i,k,A] ← (i,j,k,B,C)
```

On line 8, the algorithm reads from the array D, and on line 8, it reads and writes to the arrays D and *backtrace* on lines 9, 10 and 11. The accesses to D on line 8 actually doesn't require an array; we are reading elements sequentially. Thus, these elements can be just as easily stored in a list.

If i and k are fixed, then the accesses to D on lines 9, 10 and 11 only need to be indexed by A. In other words, the array acts as a cache at the level of the loop where i and k are constant. Thus, instead of a three dimensional array (start word, end word, and rule number), we only need a one dimensional array, of size $O(m^{r+1})$. Because of pruning, most of the array is empty, pointing to null edges. Just before i and k change, the algorithm copies the cache on to a list, which will be accessed on later iterations.

Including these changes, the algorithm now becomes:

```
1: for s=2 to n
2:   for i=1 to n−s−1
3:     let k=i+s
4:     for j=i to k−1
5:       for A=1 to #nonterms
6:         iterate through non-zero edges of the form <B,p_B> on [i,j]
7:           iterate through non-zero edges of the form <C,p_C> on [j+1,k]
8:             let p ← P(A→B C)·p_B·p_C
9:             if p > D[A] then
10:              D[A] ← p
11:      compile non-zero D[A]'s into a list with elements <A,D[A]>
```

In preliminary testing, we found the combination of caching and compiling to lists made parsing time comparable (but still lower) than using arrays for the DPA. Unlike using pruned search, caching does not change the results of parsing, so we do not report the result of experiments including and excluding this approach.

4.3.3 Models

Chen and Goodman (1998) conducted a fairly thorough examination of smoothing for language modelling. In particular, they evaluate smoothing algorithms due to Jelinek and Mercer (1980), Witten and Bell (1991), Katz (1987), and Kneser and Ney (1995), among others. The description of the smoothing techniques here shall follow the deviation and much of the terminology of Chen and Goodman (1998).

The basis of the Jelinek and Mercer (1980) and the Witten and Bell (1991) algorithms is a technique called shrinkage (also known as linear interpolation). If the training data is too sparse to properly estimate $P(X|Y)$, then Y may be broken into a number of contexts (Y,C), and the probabilities mixed together:

$$P(X|Y) = \sum_C P(C) \cdot P(X|Y,C)$$

We will use λ_C to represent $P(C)$. We shall refer to the λ's as the *mixing parameters.* We will also write out the context Y,C explicitly. For example:

$$\begin{aligned} P(X_i|X_{i-1}X_{i-2}) \approx \quad & \lambda_2 \cdot P(X_i|X_{i-1}X_{i-2}) \\ & + \lambda_1 \cdot P(X_i|X_{i-1}) \\ & + \lambda_0 \cdot P(X_i) \end{aligned}$$

However, there are a number of possible approaches to estimating the mixing parameters. Usually, the estimate of each λ depends on the associated $X|Y, C$. Many approaches to estimating λ are based upon the idea of separating *types* from *tokens* (or, alternatively, classes from instances or species from individuals). In general, the fewer types seen for each Y, C, the less the probability estimate should be trusted. Thus, the relative contribution of each λ may change as the Y's change.

On the other hand, Brants (2000) suggests using *fixed* λ's regardless of the context, arguing data may be too sparse to even estimate the λ's effectively.

4.3.3.1 Brants' Algorithm

$\tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\lambda}_3 \leftarrow 0$

`for each trigram` x_i, x_{i-1}, x_{i-2} `such that` $c(x_i, x_{i-1}, x_{i-2}) > 0$

$$d_3 \leftarrow \begin{cases} \frac{c(x_i, x_{i-1}, x_{i-2}) - 1}{c(x_{i-1}, x_{i-2}) - 1} & \text{if } c(x_{i-1}, x_{i-2}) > 1 \\ 0 & \text{if } c(x_{i-1}, x_{i-2}) = 1 \end{cases}$$

$$d_2 \leftarrow \begin{cases} \frac{c(x_i, x_{i-1}) - 1}{c(x_{i-1}) - 1} & \text{if } c(x_{i-1}) > 1 \\ 0 & \text{if } c(x_{i-1}) = 1 \end{cases}$$

$$d_1 \leftarrow \frac{c(x_i) - 1}{N - 1}$$

`if` d_3 `= max` d_1, d_2, d_3 `then`

$\lambda_3 \leftarrow \lambda_3 + c(x_i, x_{i-1}, x_{i-2})$

`else if` d_2 `= max` d_1, d_2, d_3 `then`

$\lambda_2 \leftarrow \lambda_2 + c(x_i, x_{i-1}, x_{i-2})$

`else`

$\lambda_1 \leftarrow \lambda_1 + c(x_i, x_{i-1}, x_{i-2})$

`end`

$$\lambda_1 \leftarrow \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}$$

$$\lambda_1 \leftarrow \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}$$

$$\lambda_1 \leftarrow \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}$$

Figure 4.4. Brants' Algorithm

Because it does not need extra data to estimate the mixing parameters, one advantage of the algorithm introduced in Brants (2000), henceforth the Brants algorithm) is that it is able to make more effective use of the training data. As noted above, this is one of the most accurate POS taggers, outperforming both Ratnaparkhi (1996) and Brill (1995). Because of the flatness of NEGRA trees, techniques useful for POS tagging may presumably carry over to parsing.

A novelty of Brants' algorithm, shown in Figure 4.4, is that it does not require a held-out set of training data to tune the λ parameters. Rather, the λ's are estimated directly from the main training data.

4.3.3.2 Witten-Bell Smoothing

The probability distribution for Witten-Bell smoothing

$$P_{\text{WB}}(X_i|X_{i-1}...X_{i-m}) = \lambda_m P(X_i|X_{i-1}...X_{i-m}) + (1-\lambda_m) P_{\text{WB}}(X_i|X_{i-1}...X_{i-m-1})$$

In turn, we can define $P_{\text{WB}}(X_i|X_{i-1}...X_{i-m-1})$ in terms of $P_{\text{WB}}(X_i|X_{i-1}...X_{i-m-2})$ and λ_{m-1}, until we reach the last recursion, which can be defined as $P_{\text{WB}}(X_i|X_{i-1}) = P(X_i|X_{i-1})$.

The parameters λ are defined as:

$$\lambda_j = \frac{N_{1+}(\bullet X_{i-1}...X_{i-j})}{N_{1+}(\bullet X_{i-1}...X_{i-j}) + C_j \sum c(X_i...X_{i-j})}$$

In turn, $N_{1+}(\bullet X_{i-1}...X_{i-j})$ is defined as the number of unique values for X_i that occur in the context of a particular $X_{i-1}...X_{i-j}$. More formally, we may write:

$$N_{1+}(\bullet X_{i-1}...X_{i-j}) \;=\; |\{X_i : c(X_i...X_{i-j})\}|$$

Furthermore, for the algorithm as defined as Witten-Bell, we take $C_j = 1$. Collins (1999) describes a variant where C_j is a parameter to be tuned; in particular, he uses $C_j = 5$ for his lexicalized parsing experiments. This is the particular model we use here.

4.3.3.3 Modified Kneser-Ney

Kneser and Ney (1995) introduce an algorithm with the property that the marginals of the smoothed distribution match the marginals of the empirical distribution, i.e.

$$P_{\text{smooth}}(X) \;=\; \hat{P}(X)$$

Chen and Goodman (1998) introduce a variant of this, known as modified Kneser-Ney smoothing. For the sake of completeness, we include Chen and Goodman's formulae for modified Kneser-Ney smoothing. However, re-deriving the equations is beyond the scope of this dissertation. Like Brants and Witten-Bell smoothing, modified Kneser-Ney smoothing is defined recursively. However, some extra work is necessary to ensure that the marginal distributions do indeed match the empirical distribution, making the main definition a bit more complicated than the previous distributions:

$$\begin{aligned} P_{\text{KN}}(X_i|X_{i-1},...,X_{i-m}) \;=\; & \frac{c(X_i...X_{i-m}) - D(c(X_i...X_{i-m}))}{c(X_{i-1}...X_{i-m})} \\ & + \gamma(X_{i-1}...X_{i-m}) \cdot P_{\text{KN}}(X_i|X_{i-1}...X_{i-m+1}) \end{aligned}$$

Where $D(c)$ is defined as:

$$D(c) \;=\; \begin{cases} 0 & \text{if } c=0 \\ D_1 & \text{if } c=1 \\ D_2 & \text{if } c=2 \\ D_3 & \text{if } c\geqslant 3 \end{cases}$$

And, further, the function $\gamma(X)$ is defined as:

$$\gamma(X_{i-1}...X_{i-j}) \;=\; \frac{D_1 N_1(X_{i-1}...X_{i-j}\bullet) + D_2 N_2(X_{i-1}...X_{i-j}\bullet) + D_2 N_2(X_{i-1}...X_{i-m+j}\bullet)}{c(X_{i-1}...X_{i-j})}$$

With, finally, the constants D_1, D_2, D_3 and Y set to:

$$\begin{aligned} Y &= \frac{n_1}{n_1 + 2n_2} \\ D_1 &= 1 - 2Y\frac{n_2}{n_1} \\ D_2 &= 2 - 3Y\frac{n_3}{n_2} \\ D_3 &= 3 - 4Y\frac{n_4}{n_3} \end{aligned}$$

4.3.3.4 Parsing with Markov Grammars

To this point, we have discussed various smoothing algorithms without detailing how they are relevant for our purposes. Although all three approaches above are designed to be used for n grams or HMMs, they can be easily converted for use in PCFGs. We simple add the parent node into the context. In other words, instead of

$$P(X_i|X_{i-1}X_{i-2})$$

	Prec	Recall	F-Score	Avg CB	0CB	$\leqslant$2CB	Cov	Time
Baseline	68.6	73.6	73.1	0.72	64.4	91.2	98.0	3.3s
Beam, no smoothing	70.9	74.2	72.6	0.72	63.7	91.4	97.0	2.6s
Brants	75.5	75.8	**75.7**	0.56	68.1	93.9	99.4	20.6s
Kneser-Ney	74.4	76.3	75.3	0.60	67.9	93.4	95.2	13.8s
Witten-Bell	75.1	75.2	75.1	0.56	69.5	93.1	98.9	16.5s

Table 4.6. Results with smoothing

We use

$$P(X_i|PX_{i-1}X_{i-2})$$

Where the X_i's are all children nodes and P is the parent node. This essentially gives us a 4 gram model. The Witten-Bell and Kneser-Ney are defined for all n grams, and the Brants algorithm can be trivially extended from the 3-gram case to the 4-gram case. This formulation is easy to use, but, as we shall see, it is problematic for some of the smoothing approaches.

Like with the sister-head parser, we have a special STOP node to indicate the end of a constituent. We also treat the first child as a special case, and not subject to smoothing. While this does appear like an arbitrary choice, it does allow us to use the same parsing algorithm as the sister-head parser.

4.3.4 Method

Once again, we use the same data set as in previous experiments. The variable width beam is set to $\frac{1}{4000}$. We ran three set of experiments. The first set tested all three smoothing algorithms with a single-pass CYK-like parsing algorithm. The model that served as the baseline was the best performing parser from Section 4.2, which included all re-annotations. The second set of experiments used multipass parsing with two passes.

The final set of experiments was designed to more closely study the effect of each annotation change from Section 4.2. It is worth asking if these re-annotations worked because they were intrinsically useful, or if they only helped to solve a sparse data problem. The underlying hypothesis here is that the smoothing algorithms will help overcome any sparse data problems, showing where the annotation was or was not useful. We test this hypothesis with each of the smoothing models. For this set of experiments, we use single-pass parsing.

4.3.5 Results

The results of the first set of experiments, which use a normal single-pass parser, are presented in Table 4.6. The baseline model in this case is the exhaustive parser. The next line shows the results of using a beam search without adding smoothing. The next three lines show the effect of the various smoothing algorithms. The Brants algorithm gives the highest result in the first set, with an F-score of 75.7. In addition to the standard evaluation metrics we have been using, Table 4.6 also includes the average time taken to parse a sentence. While using a beam search made parsing faster, introducing smoothing slows down parsing times considerably, by as much as 10 times for Brants smoothing.

Table 4.7 shows the results with multipass parsing. Again, the parser with Brants smoothing did best, with an F-score of 76.0. Multipass parsing was considerably faster than in the single-pass case (Brants smoothing was as much as 4 times faster). However, coverage went down for all the multipass parsers.

	Prec	Recall	F-Score	Avg CB	0CB	$\leqslant$2CB	Cov	Time
Baseline	68.6	73.6	73.1	0.72	64.4	91.2	98.0	3.3
Beam, no smoothing	72.6	75.4	74.0	0.66	63.6	92.6	88.9	2.0
Brants	75.6	75.8	**75.6**	0.59	66.7	93.5	95.1	5.7
Kneser-Ney	74.2	76.1	75.2	0.62	65.6	93.3	91.9	4.9
Witten-Bell	75.5	75.5	75.5	0.55	69.3	93.3	94.5	8.3

Table 4.7. Results with smoothing and multipass parsing

	No smoothing	Brants	Kneser-Ney	Witten-Bell
None	70.3	72.3	72.6	72.3
Co-ordination Copying	72.7	75.2	75.4	74.5
NP Case Marking	73.3	76.0	76.1	75.6
PP Case Marking	73.2	76.1	76.2	75.7
SBAR Marking	73.1	**76.3**	76.0	75.3
S Function Removing	72.6	75.7	75.3	75.1

Table 4.8. Replicating the re-annotation experiment with beam search and smoothing

The results for the third set of experiments are listed in Table 4.8, displayed with cumulative modifications as in Section 4.2. Overall, the best performing parser is again the Brants parser, but without the GF-stripping modification. It achieves the highest performance yet of an unlexicalized parser, with an F-score of 76.3.

4.3.6 Discussion

The smoothing models have vindicated the use of grammatical functions. In previous sections, we saw that grammars with GFs had lower or only marginally higher performance than grammars without GFs. However, with the addition of smoothing, the GF parser does dramatically better. Hence, we may conclude that earlier GF models were suffering from sparse data problems.

The extent of the improvement due to smoothing was dramatic enough that the scores of the GF parser were higher than any lexicalized parser from Chapter 3. This does not imply that lexicalization could not improve the performance of the GF models. Rather, it shows how introducing additionaly linguistically-motivated annotation can be as helpful or, indeed, more helpful than standard knowledge-lean approaches such as lexicalization.

Eschewing lexicalization does have its problems. Because the use of lexical features is so prevailent, there has been little research on the use of smoothing with unlexicalized grammars. To our knowledge, the present work is the only one which investigates this approach. It turns out that lexicalization has a dramatic effect on parsing efficiency. Lexicalization together with beam search was sufficiently fast for the parsers of Chapter 3. As mentioned above, this was not the case for the GF parsers. Parsing times became reasonable only after the inclusion of caching. Caching improved parsing performance by a factor of 30 or more, without any loss in accuracy. Presumably, this approach could also be applied to lexicalized parsing.

In contrast to caching, the benefits of multipass parsing were not as clear, at least with the grammars we used. While parsing was about 2 times faster, accuracy and coverage were both slightly lower. The choice of grammar used in the first pass of parsing no doubt has a large impact on the overall performance of multipass parsing. It would be informative to experiment with other grammars on the first pass. We leave this to future research, only noting that the unsmoothed no GF PCFG grammar has higher coverage and is still quite fast to parse exhaustively.

Regardless of the problems with multipass parsing, there are a number of interesting facts to learn about the choice of smoothing algorithm. Perhaps most striking is the difference in performance across smoothing algorithms. Modifications to the grammar model which result in large change in performance tend to be stable across algorithms, but there is quite a bit of variation in smaller incremental changes. For example, copying the GF of co-ordinated categories boosts F-scores by about 3 points, irrespective of the smoothing algorithm. On the other hand, marking SBARs or the case of PPs led to relatively small changes in the F-score (usually 0.1 or 0.2 points), but the change was upwards for some smoothing algorithms, and downwards for others.

The variance across smoothing algorithms is relevant as results in parsing (as well as other tasks) are often presented with just one choice of smoothing algorithm, and it is difficult to decipher if incremental improvements are due to the choice of linguistic features, or due to the choice of the particular smoothing algorithm. This is most worrying in work such as Charniak (2000) which even fails to specify exactly what approach to smoothing was used. The variability of smoothing is a concept which the language modelling community has come to appreciate (Chen and Goodman, 1998; Rosenfeld, 1999), and it is one that the parsing community should come to accept, as well.

It is not the case that all the difference between smoothing algorithms were random. One of the most interesting trends is the performance of the Brants algorithm versus the Kneser-Ney and Witten-Bell approaches. In almost all conditions, the Brants algorithm gives the best performance with Kneser-Ney in second, and Witten-Bell in third.

In some respects, this is not be surprising: the Brants algorithm was developed for POS tagging, whereas both Kneser-Ney and Witten-Bell were developed for language modelling. The event space in unlexicalized parsing bears a closer resemblance to POS tagging than language modelling. We hypothesize that one aspect of the Brants algorithm which makes it more suitable to unlexicalized parsing is that the mixing parameter λ remains constant in all contexts. This is clearly unsuitable for language modellings: in text, more frequent contexts are less likely to suffer from sparse data problems. With the much smaller set of labels in unlexicalized parsing or POS tagging, apparantly pooling the estimates for the mixing parametres provides better results.

It should be noted that unlexicalized parsing with GFs differs in some important respects from the tag set used by Brants. Most notably, POS tags have a relatively 'flat' distribution, whereas node labels which include GF labels follow a Zipfian distribution, just like text. Nonethless, the Brants algorithm is apparently able to cope with a Zipf-distributed tag set.

The modified Kneser-Ney algorithm also performed quite well, slightly better than the Witten-Bell appraoch. This parallels results in language modelling (Chen and Goodman, 1998). We pin the difference on the fact that Kneser-Ney is a more modern algorithm which has managed to take into account successful characteristics of other approaches to smoothing. It is surprising, though, that Kneser-Ney did so well despite the fact we broke the main asusmption made by the algorithm. Recall that Kneser-Ney ensures that the smoothed marginals match the empirically observed marginals. In our fourgram approach to parameterizing the grammar model, we mix the parent node in with the previous sisters. But the parent node has a differnet marginal distribution than the previous sisters because the parent is always a nonterminal whereas the previous sisters may be also be POS tags.

There may also be a problem with the marginals of the previous sisters. In an n-gram trained on free text, every word in the training data will occupy every position in the n-gram history. This is not necessarily true of context-free rules. Rules are short and tend to begin or end with a head word and/or a function word. Both the problem with the parent and with the previous sister marginals might be overcome by fixing the marginals for each position in the context. However, we leave further investigation of this to future research.

Turning our attention to the second set of experiments, it is interesting to see that the GF-stripping operations did not help with the smoothed grammars. In Section 4.2, we found that it was helpful to remove GFs from S nodes. As this operation produced mixed results with smoothing, we conclude that the models from Section 4.2 which *included* GFs on S nodes suffered from sparse data, and removing the GFs resolved this problem.

Overall, smoothing is useful for GF parsing. We have the highest result so far from an unlexicalized parsing, 76.2, which is in fact better than the lexicalized sister-head parser.

4.4 Verb Final and Topicalization Constructions

In Section 3.6 we took a detailed look at verb-final constructions with the sister-head parser. Here, we will replicate this analysis for the smoothed GF parser and we will also present a similar analysis for another construct found in German and not in English: topicalization (cf. Section 1.1.1). In main clause constructions, the verb is in the second position of a S rule, and the subject is usually in the first position (see Example 1.2).

Example 4.1.

	Ich	*esse*	*Schinken*	*in*	*dem*	*Haus*
	I	eat	ham	in	the	house
	NP	V	NP	PP		
position	I	II	III			

However, a modifier can be *topicalized* and occupy position I. In these cases, the subject moves to position III, using the verb an an axis. Other complements and modifiers come after the subject, as in Example 1.3.

Example 4.2.

	in	*dem*	*Haus*	*esse*	*ich*	*Schinken*
	in	the	house	eat	I	ham
	PP			V	NP	NP
position	I			II	III	

For formal grammar writers, topicalization (and flexible word order in general) has been the source of much research, and a number of techniques have been devised to handle this phenomenon. These techniques include movement, soft constraints in LP/ID rules and topological fields.

However, the situation for treebank grammars is slightly different. In cases when the subject moves from position I to position III are both to be found in the treebank grammar. Presumably, this ought not to make parsing much harder, as the grammar can learn both orders from the treebank.

4.4.1 Method

We proceed in a manner similar to Section 3.6, reporting both standard F-scores, and a weighted F-score measure that attempts to remove the influence of longer and more complicated sentences. We test results using four parsers. Two of these four parsers are the best performing unsmoothed model and the best performing smoothed model. One of these parsers includes the SBAR (verb-final clause) marking modification, the other does not. To round out the comparison, we ensure that models both with and without the SBAR marking modification are included in both the unsmoothed and smoothed cases.

	SBAR?	ALL	VF	NOVF	TOPIC	NOTOPIC
Avg. # of nodes		7.5	11.2	6.4	8.9	6.5
Standard F-Score	×	72.7	68.1	75.0	71.8	73.6
		72.6	67.8	75.0	72.2	72.9
Weighted F-Score	×	72.7	68.3	73.7	72.0	72.8
		72.6	68.7	73.7	72.4	72.1

Table 4.9. Performance of the unsmoothed model on various syntactic constructions

	SBAR?	ALL	VF	NOVF	TOPIC	NOTOPIC
Avg. # of nodes		7.5	11.2	6.4	8.9	6.5
Standard F-Score	×	76.1	72.8	77.9	75.1	77.2
		76.3	73.2	77.8	75.6	76.9
Weighted F-Score	×	76.1	74.6	76.8	75.9	76.4
		76.3	75.2	76.8	76.0	76.2

Table 4.10. Performance of the smoothed model on various syntactic constructions

4.4.2 Results

We show the results in Table 4.9 for the parser without smoothing, and Table 4.10 for the parser with smoothing. The 'SBAR?' column indicates if the model in question contains the SBAR re-annotation. Most of the other entries should be self-explanatory. Once again, though, there are too many results to discuss all of them in detail, so we will simply point out some of the key findings. Just as in Section 3.6, we find sentence with 'special' constructions have lower F-scores. Adding the SBAR annotation made little difference in the performance non-verb-final (NOVF) sentences in both the smoothed and unsmoothed grammars. In the unsmoothed grammar, this annotation led to mixed results in all conditions except the NOVF case, where performance was unchanged. In the smoothed grammar, on the other hand, it improved the performance in the VF condition.

Oddly enough, the unsmoothed parser did about 1 point better when the subject was not in position I (TOPIC) than when it was (NOTOPIC), despite the fact that sentences are longer in the TOPIC case. The smoothed grammar was more accurate in both the SUBJ and NOSUBJ conditions, but the relative performance swapped, with the NOTOPIC case giving the higher result.

4.4.3 Discussion

It appears that verb-final clauses are almost as difficult for the unlexicalized GF parser as for the sister-head parser. In Section 3.6, we hypothesized that parsers ought to have some special mechanism for dealing with these constructions. The SBAR annotations did improve the smoothed grammar's performance on verb-final constructions, but apparently not enough to close the gap with non-verb-final constructions. However, adding this annotation also made the weighted performance of both parsers in the TOPIC condition comparable to their weighted performance in the NOTOPIC condition (although partly by decreasing the F-score in TOPIC condition). It is interesting that, for both parsers, the difference between TOPIC and NOTOPIC is much smaller than the difference between VF and NOVF. Part of this might be explained by the smaller difference in average sentence lengths, but the change is impervious to weighting, which ought to account for part of the sentence length effect. We conjecture this is because a fronted PP has less attachment ambiguity than one which occurs in the verb's argument field. Another possibility is that (accorinding to some dependency grammar theories), verb-final clauses involve crossing dependencies, whereas topicalization does not.

Based on our initial experiments in Section 4.1, it seemed as if horizontal Markovization was a general technique, and not specific to the Penn Treebank. But our experiments here suggest that Markov grammars have some difficulty in modeling flexible word order constructions. Given that the parsers have a harder time with verb-final constructions, a possible solution is to give a better treatment of long-distance dependencies. This wouldn't help with topicalization, however. There, the problem may be that the Markov histories are not long enough. If we are considering which node to add to the partial rule S $\rightarrow$ NP-OA V NP-SB ..., the probability of addding an accusative object would be too high: the existing NP-OA is lost in the Markov history, and we could not tell if it was an object or a modifier which had been fronted.

4.5 Conclusion

One of the goals of this chapter has been vindicated: a finely tuned grammatical function parser performs better than the fully lexicalized parsers of Chapter 3. Unfortunately, we did not succeed in matching the performance of the GF parser from Section 3.2. However, the coverage of the GF parsers here was much higher.

Overall, we found that Markovization and smoothing help overcome coverage problems, and increase performance. Lexical sensitivity also helps with low-coverage problems at the cost of reducing performance, although some of this lss can be overcome by using a smart suffix analyzer when sparse data is not a problem.

While (horizontal) Markovization worked well in NEGRA, other techniques which have been shown to be useful for English, including higher-order vertical Markovization, appear to be specific to the annotation style of the WSJ, and do not generalize to NEGRA.

The two main results are (i) that Markovization may not be helpful in handling phenomena of flexible word order, such as the subject movement we saw in Section 4.4 (although difficulties with the evaluation precludes us from making a claim with any degree of certainty); and (ii) that including more knowledge about things such as noun declension is one of the factors that allow the unlexicalized parser to be competitive with the lexicalized parser.

This second point brings us back to our main argument: that linguistic features play an important role in languages with a rich morphology, and we cannot depend on lexicalization alone. For good measure, it is interesting to note that many of the features we annotated dealt with case, and would not even be relevant for an English parser.

Chapter 5

Parsing with Attributes

In Chapter 4, we found that paying attention to grammatical functions in NEGRA leads to more accurate parses. We argued the benefit from GFs is due to their ability to model syntactic phenomena, including aspects of the German case system. The success of the GF model elicits an additional question: are there further gains to be had by including additional linguistic information which is typicalled excluded from a PCFG model? In this chapter, we investigate two such phenomena: morphology and long-distance dependencies (LDDs). Let us consider morphology first.

A full model of morphology might require lexicalization in the spirit of the models discussed in Chapter 3. However, in Chapter 4 we argued that the increase in accuracy when including GFs was in part because they encode some lexical information. In other words, lexical parameterization and 'lexical' attributes (like some GFs) both carry some of the same information. Strictly speaking, however, the two choices are not equivalent. In theory, lexicalization also contains more abstract information such as selectional preferences. However, it is worth investigating if there is some benefit to including more lexical features. Based upon the success of including some basic case information in the models of Chapter 4, we focus our attention on the morphosyntactic attributes of case, gender, number and person.

Although this is a small set of attributes, including them in our grammar model is not a trivial task. There are two major difficulties. First, there is no suitable training data available. While the NEGRA corpus is annotated with these attributes on some sentences, most of the corpus contains no such information. A second problem is that even the small set of information we wish to add may introduce sparse data problems, and would hence require a different approach to parameterizing the grammar.

Parsing with LDDs is somewhat easier: as non-local dependencies play an important role in German syntax, these dependencies tend to be annotated in treebanks, including NEGRA. However, we show that the simplest PCFG model which can accounts for LDDs is unable to derive much benefit from them. Only after some re-annotation is a PCFG model able to profit from including a model of LDDs.

This chapter is organized as follows. In Section 5.1, we describe the construction of the morphologically-tagged training data. In Section 5.2, we then test a model trained upon this data. We then offer a further refinement of this model in Section 5.3. In Section 5.4, we turn our attention developing a parser that can account for LDDs. In Section 5.5, we provide some further evaluation of the LDD parser. Finally, we offer some concluding remarks in Section 5.6.

	Symbol	Description
Case	Accusative	Akk
	Dative	Dat
	Genitive	Gen
	Nominative	Nom
Gender	Feminine	Fem
	Masculine	Masc
	Neuter	Neut
Number	Pl	Plural
	Sg	Singular

Table 5.1. List of the morphological tags

5.1 Semi-automatic Morphology Annotation

This section describes the construction of the morphologically tagged corpus which is used in later sections of this chapter. Section 5.1.1 describes the basics of constructing the corpus, and provides an evaluation and error analysis of the automatic tagging. Based on the results of the evaluation and error analysis, Section 5.1.2 introduces techniques to increase the overall accuracy of the morphological tagging and to remove unwanted tags. Finally, Section 5.1.3 shows how to correctly and consistently augment a context-free grammar with morphological tags. A discussion of the actual parameterization of such a grammar, however, is left until Section 5.3.

5.1.1 Building a morphologically tagged corpus

Two auxiliary morphological taggers are used to build the morphologically tagged corpus. The taggers, Morphy (Lezius et al., 1998) and DMM (Lorentz, 1996), take the words of NEGRA corpus as input, labelling each word with its possible morphological features.

Data

Because we intend to tag the whole corpus, we do not use the normal split into training, development and test sets. However, without a proper test set, we have no way to evaluate the accuracy of the morphological taggers. Fortunately, the first 6390 sentences in NEGRA do have morphological information, so it is possible to use these sentences as a morphological test set (distinguished from the normal test set used to evaluate parsing results). The sentences which comprise the morphological test set are also part of the syntactic training set. Later in this chapter, we will freely use all of the syntactic training set to train a morphologically-aware grammar model. One might argue against this approach on philosophical grounds, but recall that we are not developing a model of German morphology. Rather, the morphological test set is simply a way to measure how well the existing taggers perform.

The format of the morphological tags in the generated corpus is a simplified version of what is already used in the morphologically annotated part of NEGRA. To describe how the format is simplified, we need to clarify some terminology. What we are calling a morphological tag is the complete morphological description of a word (as far as the NEGRA annotation permits). We say that each tag has a number of components. The components we consider are case, gender, number and person. The particular values we use are fairly straightforward, but for reference they are listed in Table 5.1. For example, a possible tag for the word *Vater* is Masc.Sg.Nom, and the components are Masc, Sg and Nom.

We simplify the NEGRA tagset by ignoring some tag components. These components are for inflections such as verb conjugation which the error analysis in Chapter 4 suggested were less important than the ones we use. All of these extra components are stripped from the corpus.

	Morphy		DMM		Combined	
	Partial	Exact	Partial	Exact	Partial	Exact
Substantives	67.1	66.4	92.0	90.4	94.8	93.7
Names	53.8	0.0	65.3	64.9	71.3	64.9
Verbs	98.3	98.2	98.6	98.4	99.0	98.9
Adjectives	89.9	89.6	--	--	89.9	89.6
Articles & Pronouns	99.3	93.7	99.3	93.7	99.4	97.8
Prepositions	73.0	53.2	97.1	78.0	97.6	86.6
Overall	79.9	70.8	88.9	81.6	95.6	91.8

Table 5.2. Accuracy of morphological tagging

The NEGRA format is not directly compatible with those of DMM and Morphy. Therefore, the output of the taggers must be converted to the NEGRA format. The conversion is mostly straightforward, but several non-trivial cases are discussed below in the error analysis section.

To make use of the tagged versions of the corpus, we must re-align words in the tagged versions with the matching words in the original. For the most part, this can be done by simply scanning the output of the taggers sequentially. However, it is not the case that the nth word in the tagged corpus matches the nth word in the original corpus. Three idiosyncrasies of the taggers need to be taken into account. First, DMM outputs two entries for words like *Dr.* which end with a period. The second entry must be ignored. A second problem is foreign characters like *é* confuse Morphy. All words containing such letters are therefore ignored. A final problem is that Morphy splits words containing an apostrophe. This would make sense for analyzing contractions like *geht's*, but NEGRA already splits contractions into two words. Because Morphy's extra splits are often nonsensical, we skip all of the entries from these words. The last two idiosyncrasies sometimes occur together, in words like *Bahá'u'lláh*.

Evaluation

We evaluate the output of the taggers against the gold-standard annotated tags in NEGRA. We use two metrics to measure how often morphological taggers agree with the annotation: exact matches and partial matches. We say that a tag matches the gold-standard exactly when each of their components match, and they have exactly the same components. We say a tag matches partially if all components present in both tags match, but one or both tags may have extra components. For example, the preposition-article *im* is usually tagged in NEGRA as Dat.Sg. An aggressive tagger may instead suggest Dat.Neut.Sg. The tags Dat.Sg and Dat.Neut.Sg partialy match because the Dat and Sg components are the same, and the extra Neut is ignored. The tags do not have an exact match because of the extra Neut tag.

Both taggers output a number of hypotheses per word. We do not attempt to disambiguate the hypotheses and pick one as 'best'. Rather, we consider the morphological tagger to have made an exact match if *any* hypothesis it suggests has an exact match with the gold annotation in NEGRA (and likewise for partial matches).

Results

The results of the morphological taggers are shown in Table 5.2. The overall results obtained when combining the output of the two taggers is 95.6% on partial matches and 91.8% on exact matches. Many parts of speech are done particularly well, including verbs, articles and pronouns. Morphy does well on adjectives, but the results of DMM cannot be compared because it uses a different theory of adjective inflection. Morphy did surprisingly poorly on substantives, and both had difficulties with proper names. Overall, the results are high, but given how easy the task is, it is surprising the results are not higher. Moreover, the tags for some parts-of-speech appear to be problematic. It would be insightful to determine why some categories perform so poorly.

Error Analysis

We investigate these troublesome tags with an error analysis. This is interesting not only for our own purposes, but automatically tagging the annotated subset of a large corpus gives some insights into how well the morphological taggers do on "real" German. Some of the most frequent causes of errors appear to be:

- Annotation errors. In some cases, the NEGRA annotation is simply incorrect. In many other cases, the annotated words have incomplete (but correct) tags. This lowers exact matches, while partial matches are still high. This problem is particularly noticeable with prepositions.
- Theoretical mismatches. As noted above, DMM has a different approach to analyzing adjectives. It labels articles with adjective endings they allow, but does not analyze adjectives themselves. There are other problems. For instance, *andere* ("other") is always an adjective in Morphy/DMM. In NEGRA, *andere* is only tagged as an adjective in noun phrases like *die anderen Programme* ("the other programs") where it modifies a noun. When *andere* is the head of a noun phrase (as in Example 5.1 below), it is tagged as a pronoun:

	Für	*andere*	*stellen*	*sie*	*Weichen*
Example 5.1.	For	others	set	they	switch
	"They set the course for others"				

- Coverage problems. Both taggers have poor coverage on proper nouns. They are much better with other open-class words, but Morphy also has significant difficulty with substantives. It appears as if Morphy has difficulty finding the parts of compound nouns. While DMM appears to be much better at analyzing compound nouns, for reasons unclear to us, it has difficulties with others such as *Vogelgezwitscher* ("bird twitter") and *Mitsprache* ("co-determination"). Some of the coverage failures can be classified into a few common classes:
  - Hypenated words, and other strange compounding, including numerical compounding. For instance: *21jäahrigen* ("21 year olds")
  - Borrowed foreign words, such as *Jeans, Tattoos, Techno, Groove* and *Sunnis*
  - Abbreviations, such as *Dr., Tel., u.*
  - Spelling mistakes, like *Wahrnehumng* (*Wahrehmung*, "perception"), are not very frequent, but still affect some words.
- Part-of-speech errors. At times, the morphological tagger may mistake the POS of a word, giving an incorrect analysis. For example, DMM treats *Kabelfernsehen* ("cable television") and *betroffene* ("affected") as verbs.

5.1.2 Morphology and context

Though we have described how to generate a morphologically-tagged corpus, the corpus requires further changes before it can be used by the parser . There are two problems with the corpus as it stands. First, as we saw above, the taggers have poor coverage with some parts-of-speech, especially nouns. This means that they will either suggest no tag or an incorrect tag for many nouns. A second problem is that German morphological suffixes are ambiguous, and can only be disambiguated when context is taken into account. Unfortunately, all the morphological taggers ignore context. For example, they will not ensure that subjects agree with verbs. Parse trees provide information useful to moderate the extent of both these problems.

Constraint #	Parent	Child	Rule
1	NP, PP, MPN, AP, CAP	Two declinables	Exact match
2	PP	Preposition & declinable	Partial match
3	S	Verb and subject	Partial match
4	CNP	Two declinables or NPs	Case matches

Table 5.3. Constrains to eliminate incorrect morphological tags.

To overcome coverage problems, we can add additional hypotheses when either DMM or Morphy fail to give any analysis of a noun. Without any contextual information, we would need to add 24 hypothesis (4 cases $\times$ 3 genders $\times$ 2 numbers). As noted in Chapter 4, trees are often annotated for case, and using this information means that only gender and number are undetermined. Thus, we only need to suggest 6 hypotheses. For nouns inside PPs which are headed by a case-ambiguous preposition, there are 12 possible hypotheses (only 2 of 4 cases are eliminated). We only suggest additional tags for common nouns. Pronouns already have high coverage, and coverage is so low for proper nouns that this fall-back technique would be invoked too often to be useful.

Let us now turn to the problem of superfluous tags. Consider the following sentence:

Example 5.2.

Der	*Alltags*	*steht*	*in*	*Zentrum*	*des*	*Films*
The	everyday life	remains	in	center	the	film

"Everyday life is the main theme of the movie"

Example 5.3 shows the same sentence as tagged by DMM. The sentence is depicted vertically, with the words on the right. To the left of each word is a list of morphological features, with the '...' denoting there are more tags than we have space to show.

Example 5.3.

Der	Nom.Masc.Sg	Gen.FemSg	Dat.Fem.Sg	Gen.Neut.Pl	...
Alltags	Nom.Masc.Sg	Dat.Masc.Sg	Akk.Masc.Sg		
steht	Sg.3	Pl.2			
im	Dat.Sg				
Zentrum	Nom.Neut.Sg	Dat.Neut.Sg	Akk.Masc.Sg		
des	Gen.Masc.Sg	Gen.Neut.Sg			
Films	Gen.Masc.Sg				

The amount of ambiguity varies between words, with *im* having only one possible tag, whereas *Der* having more than four. But much of this ambiguity is spurious. *Der Alltags* is an NP, and we know the morphological features of *Der* ought to match those of *Alltags*. Because *Alltags* cannot be genitive or dative feminine, we conclude these cannot be the correct tags for *Der* in this instance. Furthermore, *Der Alltags* is the subject of the sentence, and so *steht* cannot be conjugated for the second person plural. *Zentrum* is clearly not nominative or accusative; it is a dependant of the dative preposition *im*.

We formalize these intuitions into an algorithm by defining a set of contraints stating how extra tags are eliminated. Given a rule $P \rightarrow C_0\, C_1 \ldots C_n$, the constraints force pairs of adjacent daughters C_i and C_{i+1} to have a consistant set of morphological tags. Consistancy is guaranteed by eliminating tags from C_{i+1} for which a partial or exact match cannot be found from the list of tags for C_i, and vice versa. The use of a partial or exact match depends upon P, C_i and C_{i+1}. Table 5.3 shows the full list of the contraints. The first column shows the values of P, the second, the values for C_i and C_{i+1} and the third, the type of match to be enforced. For any triplet of P, C_i and C_{i+1} not listed in the table, no constraint is applied and hence no tags are eliminated.

Some of the terms used in the table deserve explanation. In PPs and various types of adjectival and noun phrases, we enforce exact matches when both daughters are *matching declinables*. For us, a matching declinable is a noun, name, article, pronoun, adjective, adjectival phrase or co-ordinated adjectival phrase. We do not count NP daughters of NPs or PPs as matching declinables, as they are often modifiers, potentially with a different case (usually genitive). Most constraints use partial matches and exact matches as described in Section 5.1.1. The only exception is the rule for co-ordinated NPs (CNPs). Because two co-ordinate daughters might have a different gender, or even number, we only enforce that the case must be the same.

The first three constraints are invoked by the sentence from Example 5.3. Applying these constraints, we get the following set of disambiguated tags:

Example 5.4.

Der	Nom.Masc.Sg
Alltags	Nom.Masc.Sg
steht	Sg.3
im	Dat.Sg
Zentrum	Dat.Neut.Sg
des	Gen.Masc.Sg
Films	Gen.Masc.Sg

Der and *Alltags* only have one tag in common, Nom.Masc.Sg, and this is the only tag left after applying Constraint #1 on these two words. The tag Pl.2 is eliminated from the verb *steht* when Constraint #3 is run with *steht* and the noun phrase *Der Alltags*. Contraint #2 is invoked with *im* and *Zentrum*, eliminating the nominative and accusative readings of *Zentrum*. Contraint #1 is again used for *des* and *Films*, and the only tag which survives elimination is Gen.Masc.Sg.

This example may be slightly misleading because all the tags are perfectly disambiguated, and no word is left without a tag. This is not always so. If the gender of an unknown nominative plural noun cannot be guessed, the article (*die*) provides no clues on how to disambiguate, and the constraints leave more than one possible tag on the word. Whenever the morphological taggers miss the 'correct' tag for a word, the constraints may eliminate all the tags of that word because no tag in the context matches the incorrect tags. While these two problems remain, the approach does eliminate many incorrect tags. Now we turn to evaluating how well tag guessing and the tag elimination constraints both work.

Evaluation It is possible to use the same approach as in Section 5.1.1 to evaluate the tag guessing and the constraints. Just as in Section 5.1.1, we report partial and exact matches. To discern how many contextually incorrect hypotheses are being pruned by the constraints, we also report the percentage of words with exactly one hypothesis which was an exact match, and the average number of hypotheses per word.

Results and Discussion The results are shown in Table 5.4 for three conditions: the tagged corpus as it stood in Section 5.1.1 (Original), after contextual cues were used to add extra morphological tags (Extra Tags), and after pruning contextually inappropriate tags (Pruned). Adding extra tags increases the accuracy, at the cost of adding some extra hypotheses. Pruning, in turn, cuts the number of hypotheses in half while decreasing the overall accuracy as measured by partial and exact matches. There were a number of factors contributing to the fall in accuracy. Most importantly, pruning uses exact matches, which eliminates some tags which happened to agree with the gold standard by chance. This can occur because NEGRA at times uses the token * to denote an ambiguous component. For example, in the first sentence of NEGRA, *aller* from *aller Musikbereiche* is tagged as *.Gen.Pl. Using our notation, we eliminate the * from the gold standard, and assume the tag is Gen.Pl. This happens to be among the hypotheses suggested by the taggers. However, the gender of *Musikbereiche* is known to be masculine, hence the tag is Masc.Gen.Pl. Exact matching thus eliminates the 'correct' entry of Gen.Pl for *aller*.

	Original	Extra Tags	Pruned
Partial matches	95.6	97.4	95.2
Exact matches	91.8	93.8	89.1
Exact matches with one hypothesis	19.4	15.4	54.8
Hypotheses per word	5.7	6.2	2.6

Table 5.4. Taking context into account: accuracy and brevity of the hypotheses.

While pruning lowers the accuracy of the tagging, it reduces the number of hypothesis and increases the number of tags which are perfectly disambiguated. Without pruning, only $\frac{1}{6}$ to $\frac{1}{5}$ of all tags were correctly disambiguated, i.e. there was only one tag, and it was an exact match. With pruning, more than $\frac{1}{2}$ of all tags were correctly disambiguated. Our empirical experience suggests that reducing the number of hypothesis and increasing disambiguated hypotheses are together more important than the small decrease in accuracy due to pruning.

5.1.3 Morphology and grammar rules

We are almost in a position where we can use the morphologically-tagged corpus as training data for a parser. But, to this point, all the morphological tags have only been associated with words. This is insufficient for parsing. We must propagate the tags up the tree, so that syntactic categories are associated with morphological features.

To illustrate why this is so, consider the simple rule S $\rightarrow$ NP-SB VVFIN NP-OA. If we take the simple approach (similar to Chapter 4), and add morphological tags to POS tags, we may get a rule such as S $\rightarrow$ NP-SB VVFIN.Sg.2 NP-OA. Now, if we wish to enforce subject-verb agreement, we need the verb's morphological tag to 'communicate' with the subject's. To do this, we must add morphological information onto the recursive category NP-SB. In this case, the rule becomes S $\rightarrow$ NP-SB.Sg.2 VVFIN.Sg.2 NP-OA. Such communication is necessary for other nodes, as well. Thus, in general, we need to annotate the parse trees with the morphological information.

To propagate the morphological tags up the parse tree, we use the familiar technique of picking one child as the head, and projecting its attributes onto the parent. If the tag of a word is unambiguous, we are done: each node in the tree is annotated with a morphological tag.

When the tags are ambiguous, however, we need to do some extra work. We need to ensure that sequences of tags are grammatical. Consider the accusative NP 'die Biwakierenden' (those who use temporary encampments). Biwakierenden, a low-frequency noun, is a compound of Biwak and irend. Biwaki itself is a low-frequency word, making it difficult for the morphology taggers to guess the gender of Biwakierenden. Thus we get the following possible tag sequences:

Example 5.5.

die	Akk.Masc.Pl	Akk.Fem.Pl	Akk.Neut.Pl
Biwakierendern	Akk.Masc.Pl	Akk.Fem.Pl	Akk.Neut.Pl

We want to include rules such as NP.Akk.Masc.Pl $\rightarrow$ ART.Akk.Masc.Pl NN.Akk.Masc.Pl, but eliminate rules such as NP.Akk.Masc.Pl $\rightarrow$ ART.Akk.Fem.Pl NN.Akk.Masc.Pl. We adopt the same approach as Section ?: we use hand-written rules to prune unwanted rules. In fact, the same rules are re-used here.

The basic approach is to begin at the head word (*Biwakierendern*), and create a path to the left and right for each possible tag that the head word can take. In this case, there are three initial paths (one for each choice of gender). Let us call the head child the 0^{th} node. Say the n^{th} node has t_n tags, and suppose there are p_{n-1} paths at the node $n-1$. Then, there can be as many as $p_{n-1} \cdot t_n$ nodes through the n^{th} node. The pruning rules are then evoked to remove unwanted paths. In this case, without pruning, there would be 9 paths without pruning: 3 choices for the head node, 3 for the first child to the left. However, pruning results in only 3 possible paths: the gender of the pronoun must match the gender of the head word.

One problem with this pruning approach is that in many cases, non-head children are not required to have the same morphological tags as the head child. There is an exponential blow-up in the number of paths when this happens. If there are on average p hypotheses per word, then there will be p^n paths on the n^{th} node. The problem becomes apparent when processing extremely flat rules (which have a big n), as well as rules dominating horribly ambiguous proper nouns (which have a big p). A rule with both a big n and p can easily exhaust the memory on even a relatively modern computer. However this problem is easily solved if we are using a grammar with Markov rules. Recall that, if we are at node n then an m^{th} order Markov rule 'forgets' what happened at node $n-m$. Therefore, we can 'rejoin' some paths, resulting in a total of only p^m paths. In the parsers we have been using, m is usually less then 3.

Thus, at any node, we can get a reasonable set of morphological tags, and enough of the previous tags for an m^{th} order Markov grammar, provided m is small enough. Overall, we have training data, and a way for a learning algorithm to make use of the data. Now we turn to the problem of parameterizing the model the learning algorithm will train.

5.2 Parsing with Morphological Features

Once morphological tags are inserted into a tree as described in Section 5.1.3, it is straightforward to include them in a grammar model. There is one caveat: the tags are potentially ambiguous, and the probability model needs to cope with this. In this section, we show how to handle this ambiguity, and present the results of a parser trained on the morphological data set.

5.2.1 Notation

Until this point, we have allowed the nonterminal vocabulary of our grammars to become somewhat complex by coupling extra annotations together with the nonterminal symbols. In NEGRA, an accusative object NP is given the label NP-OA. With our morphology annotation, the node label may become more complicated, like NP-OA.Masc.Akk.Sg.

Due to this coupling, the NP, OA and Masc.Akk.Sg have no meaning on their own. This is not necessarily a good assumption. Consider the OA label. Although accusative and dative NP's decline differently, they share many of the same characteristics (i.e. both are the projection of a noun, both can be modified by adjectives or clauses in the same way, both have the same rules to determine if a definite or indefinite article is to be used).

The -OA can be decoupled from the NP by treating it as a *feature* (or *attribute*) of the NP node. To make this distinction explicit in our notation, instead of writing rules as such:

$$\text{NP-OA.Masc.Akk.Sg} \rightarrow \text{ART-OA.Masc.Akk.Sg NN-NK.Masc.Akk.Sg}$$

We will instead write them as:

$$\begin{bmatrix} \text{NP} & \\ \textsc{gf} & \text{OA} \\ \textsc{morph} & \text{Masc.Akk.Sg} \end{bmatrix} \rightarrow \begin{bmatrix} \text{ART} & \\ \textsc{gf} & \text{OA} \\ \textsc{morph} & \text{Masc.Akk.Sg} \end{bmatrix} \begin{bmatrix} \text{NN} & \\ \textsc{gf} & \text{NK} \\ \textsc{morph} & \text{Masc.Akk.Sg} \end{bmatrix} \tag{5.1}$$

This follows notation commonly used in attribute-value grammars (Shieber, 1986).

5.2.2 Parameterization

As in Chapter 4, we assume the use of a Markov grammar. Excluding the GF and morphological tags for a moment, an 'event' is a partial rule application:

$$\mathrm{X} \rightarrow \ldots\, \mathrm{Y}_{i-2}\, \mathrm{Y}_{i-1}\, \mathrm{Y} \ldots \tag{5.2}$$

Such a partial rule has an unsmoothed probability of:

$$P(\mathrm{Y}_i|\mathrm{X} \rightarrow \ldots \mathrm{Y}_{i-2}\ \mathrm{Y}_{i-1}) = \frac{\#(\mathrm{X} \rightarrow \ldots \mathrm{Y}_{i-2}\, \mathrm{Y}_{i-1}\, \mathrm{Y}_i \ldots)}{\#(\mathrm{X} \rightarrow \ldots \mathrm{Y}_{i-2}\, \mathrm{Y}_{i-1} \ldots)} \tag{5.3}$$

Including the tags, a partial rule becomes:

$$\begin{bmatrix} \mathrm{X} & \\ \text{GF} & f(\mathrm{X}) \\ \text{MORPH} & m(\mathrm{X}) \end{bmatrix} \rightarrow \ldots \begin{bmatrix} \mathrm{Y}_{i-2} & \\ \text{GF} & f(\mathrm{Y}_{i-2}) \\ \text{MORPH} & m(\mathrm{Y}_{i-2}) \end{bmatrix} \begin{bmatrix} \mathrm{Y}_{i-1} & \\ \text{GF} & f(\mathrm{Y}_{i-1}) \\ \text{MORPH} & m(\mathrm{Y}_{i-1}) \end{bmatrix} \begin{bmatrix} \mathrm{Y}_{i} & \\ \text{GF} & f(\mathrm{Y}_{i}) \\ \text{MORPH} & m(\mathrm{Y}_{i}) \end{bmatrix} \ldots \tag{5.4}$$

Informally, we write the probability of this rule as:

$$P\left(\begin{bmatrix} \mathrm{Y}_{i} & \\ \text{GF} & f(\mathrm{Y}_{i}) \\ \text{MORPH} & m(\mathrm{Y}_{i}) \end{bmatrix} \middle| \begin{bmatrix} \mathrm{X} & \\ \text{GF} & f(\mathrm{X}) \\ \text{MORPH} & m(\mathrm{X}) \end{bmatrix} \rightarrow \ldots \begin{bmatrix} \mathrm{Y}_{i-2} & \\ \text{GF} & f(\mathrm{Y}_{i-2}) \\ \text{MORPH} & m(\mathrm{Y}_{i-2}) \end{bmatrix} \begin{bmatrix} \mathrm{Y}_{i-1} & \\ \text{GF} & f(\mathrm{Y}_{i-1}) \\ \text{MORPH} & m(\mathrm{Y}_{i-1}) \end{bmatrix} \right) \tag{5.5}$$

Note that Equation 5.4 corresponds to the following conditional probability:

$$P(Y_i, f(Y_i), m(Y_i)|X, f(X), m(X), Y_{i-2}, f(Y_{i-2}), m(Y_{i-2}), Y_{i-1}, f(Y_{i-1}), m(Y_{i-1}))$$

Just as we updated Rule 5.2 to Rule 5.4 to account for features, we must similar update the probability estimator:

$$\frac{\#\left(\begin{bmatrix} \mathrm{X} & \\ \text{GF} & f(\mathrm{X}) \\ \text{MORPH} & m(\mathrm{X}) \end{bmatrix} \rightarrow \ldots \begin{bmatrix} \mathrm{Y}_{i-2} & \\ \text{GF} & f(\mathrm{Y}_{i-2}) \\ \text{MORPH} & m(\mathrm{Y}_{i-2}) \end{bmatrix} \begin{bmatrix} \mathrm{Y}_{i-1} & \\ \text{GF} & f(\mathrm{Y}_{i-1}) \\ \text{MORPH} & m(\mathrm{Y}_{i-1}) \end{bmatrix} \begin{bmatrix} \mathrm{Y}_{i} & \\ \text{GF} & f(\mathrm{Y}_{i}) \\ \text{MORPH} & m(\mathrm{Y}_{i}) \end{bmatrix} \ldots \right)}{\#\left(\begin{bmatrix} \mathrm{X} & \\ \text{GF} & f(\mathrm{X}) \\ \text{MORPH} & m(\mathrm{X}) \end{bmatrix} \rightarrow \ldots \begin{bmatrix} \mathrm{Y}_{i-2} & \\ \text{GF} & f(\mathrm{Y}_{i-2}) \\ \text{MORPH} & m(\mathrm{Y}_{i-2}) \end{bmatrix} \begin{bmatrix} \mathrm{Y}_{i-1} & \\ \text{GF} & f(\mathrm{Y}_{i-1}) \\ \text{MORPH} & m(\mathrm{Y}_{i-1}) \end{bmatrix} \ldots \right)} \tag{5.6}$$

However, we cannot use this estimator directly in all cases. Because the set of morphological tags is ambiguous, there may be several values of $f(Y_i)$ and $m(Y_i)$ for each rule. It is possible to account for this ambiguity by using an expensive unsupervised training algorithm such as the EM algorithm (Dempster et al., 1977). However, we were able to devise a novel approach to estimating the probabilities which is much faster in practice.

To illustrate our approach, consider the following rule, where the 'current node' (i.e. Y_i in the rule schemas above) is a noun:

$$\begin{bmatrix} \mathrm{PP} & \\ \text{GF} & \mathrm{MO} \\ \text{MORPH} & \mathrm{Akk} \end{bmatrix} \rightarrow \begin{bmatrix} \mathrm{APPR} & \\ \text{GF} & \mathrm{OA} \\ \text{MORPH} & \mathrm{Akk} \end{bmatrix} \begin{bmatrix} \mathrm{NN} & \\ \text{GF} & \mathrm{NK} \\ \text{MORPH} & x \end{bmatrix} \tag{5.7}$$

The variable x can take values from the set $X = \{\text{Masc.Sg.Akk, Fem.Sg.Akk,}\ldots\}$. Let x_i represent the ith element of set X. For the sake for this example, suppose we wish to estimate the following probability:

$$P\left(\begin{bmatrix} \mathrm{NN} & \\ \text{GF} & \mathrm{NK} \\ \text{MORPH} & x \end{bmatrix} \middle| \begin{bmatrix} \mathrm{PP} & \\ \text{GF} & \mathrm{MO} \\ \text{MORPH} & \mathrm{Akk} \end{bmatrix} \rightarrow \begin{bmatrix} \mathrm{APPR} & \\ \text{GF} & \mathrm{OA} \\ \text{MORPH} & \mathrm{Akk} \end{bmatrix} \right)$$

Using the chain rule, we separate the morphological tag x from the rest of the rule:

$$
\begin{aligned}
& P\left(\begin{bmatrix} \text{NN} & \\ \textsc{gf} & \text{NK} \\ \textsc{morph} & x \end{bmatrix} \middle| \begin{bmatrix} \text{PP} & \\ \textsc{gf} & \text{MO} \\ \textsc{morph} & \text{Akk} \end{bmatrix} \rightarrow \begin{bmatrix} \text{APPR} & \\ \textsc{gf} & \text{OA} \\ \textsc{morph} & \text{Akk} \end{bmatrix}\right) \\
= \; & P\left(\textsc{morph} \;\; x \middle| \begin{bmatrix} \text{PP} & \\ \textsc{gf} & \text{MO} \\ \textsc{morph} & \text{Akk} \end{bmatrix} \rightarrow \begin{bmatrix} \text{APPR} & \\ \textsc{gf} & \text{OA} \\ \textsc{morph} & \text{Akk} \end{bmatrix} \begin{bmatrix} \text{NN} & \\ \textsc{gf} & \text{NK} \end{bmatrix}\right) \\
& \cdot P\left(\begin{bmatrix} \text{NN} & \\ \textsc{gf} & \text{NK} \end{bmatrix} \middle| \begin{bmatrix} \text{PP} & \\ \textsc{gf} & \text{MO} \\ \textsc{morph} & \text{Akk} \end{bmatrix} \rightarrow \begin{bmatrix} \text{APPR} & \\ \textsc{gf} & \text{OA} \\ \textsc{morph} & \text{Akk} \end{bmatrix}\right)
\end{aligned}
$$

Then we assume the morphological tag is independent of everything else, and is assigned a probability according to the distribution P_X:

$$
P_X(x) \;=\; P\left(\textsc{morph} \;\; x \middle| \begin{bmatrix} \text{PP} & \\ \textsc{gf} & \text{MO} \\ \textsc{morph} & \text{Akk} \end{bmatrix} \rightarrow \begin{bmatrix} \text{APPR} & \\ \textsc{gf} & \text{OA} \\ \textsc{morph} & \text{Akk} \end{bmatrix} \begin{bmatrix} \text{NN} & \\ \textsc{gf} & \text{NK} \end{bmatrix}\right)
$$

We refer to the overall rule estimate as $P_{\text{A-RHS}}$ (for ambiguous tag on the right hand side), and it is defined as:

$$
\begin{aligned}
& P_{\text{A-RHS}}\left(\begin{bmatrix} \text{NN} & \\ \textsc{gf} & \text{NK} \\ \textsc{morph} & x \end{bmatrix} \middle| \begin{bmatrix} \text{PP} & \\ \textsc{gf} & \text{MO} \\ \textsc{morph} & \text{Akk} \end{bmatrix} \rightarrow \begin{bmatrix} \text{APPR} & \\ \textsc{gf} & \text{OA} \\ \textsc{morph} & \text{Akk} \end{bmatrix}\right) \\
= \; & P_X(x) \cdot P\left(\begin{bmatrix} \text{NN} & \\ \textsc{gf} & \text{NK} \end{bmatrix} \middle| \begin{bmatrix} \text{PP} & \\ \textsc{gf} & \text{MO} \\ \textsc{morph} & \text{Akk} \end{bmatrix} \rightarrow \begin{bmatrix} \text{APPR} & \\ \textsc{gf} & \text{OA} \\ \textsc{morph} & \text{Akk} \end{bmatrix}\right)
\end{aligned} \tag{5.8}
$$

The second probability on the second line of Equation (5.8) can be estimated from the corpus, with one caveat. While the MORPH attribute on the LHS (Akk) is unambigous in this example, it is not guaranteed to be unambiguous. For example, prepositions ambigous between the accusative and dative may take an Akk or Dat MORPH attribute. When the MORPH attribute is ambiguous on the LHS, we make the following simplifying assumption:

$$
\begin{aligned}
& P\left(\begin{bmatrix} \text{NN} & \\ \textsc{gf} & \text{NK} \end{bmatrix} \middle| \begin{bmatrix} \text{PP} & \\ \textsc{gf} & \text{MO} \\ \textsc{morph} & \text{Akk} \end{bmatrix} \rightarrow \begin{bmatrix} \text{APPR} & \\ \textsc{gf} & \text{OA} \\ \textsc{morph} & \text{Akk} \end{bmatrix}\right) \\
= \; & P\left(\begin{bmatrix} \text{NN} & \\ \textsc{gf} & \text{NK} \end{bmatrix} \middle| \begin{bmatrix} \text{PP} & \\ \textsc{gf} & \text{MO} \end{bmatrix} \rightarrow \begin{bmatrix} \text{APPR} & \\ \textsc{gf} & \text{OA} \end{bmatrix}\right)
\end{aligned} \tag{5.9}
$$

In other words, we simply use the underlying rule. Although this removes the dependance on the MORPH attribute, the value of the attribute still plays a role when calculating the tree probability. The Akk and Dat attributes are calculated at an earlier step, when the parent is being generated.

The first probability in Equation (5.8) can be estimated from the corpus, because part of NEGRA is, in fact, annotated with morphological tags. The estimate is based on marginal probabilities of morphological attributes:

$$
P_X(x) = \frac{P(x)}{\sum_{x_i \in X} P(x_i)}
$$

Note that, in this particular example:

$$
P_X(x) \;=\; P(x|\text{Akk})
$$

If ambiguous MORPH attribute on the RHS of a rule depends on an ambiguous MORPH attribute on the LHS of a rule, we will have a different set X and hence a different probability distribution P_X for each choice on the LHS. More concretely, if the LHS is ambiguous between Akk and Dat, then in the Akk case, $X = \{\text{Masc.Sg.Akk, Fem.Sg.Akk,...}\}$.

Putting everything together, we have four possible cases:

1. There are no ambiguities in the morphological tags. We may use the standard probability distribution. We refer to this case as UNAMBIGUOUS, and to the probability estimate associated with this case $P_{\text{UNAMBIGUOUS}}$.

2. There is an ambiguity on the RHS. We use P_X to 'weight' the probability estimate, as in Equation (5.8). We call to this case as A-RHS, use the probability distribution $P_{\text{A-RHS}}$ as defined in Equation (5.8).

3. There is an ambiguity on the LHS. We call such cases A-LHS (for ambiguous left hand side), and the approximation of the probability distribution $P_{\text{A-RHS}}$. $P_{\text{A-RHS}}$ is derived by applying the simplification from Equation (5.9) to $P_{\text{UNAMBIGUOUS}}$

4. There are ambiguities on both the LHS and RHS. Such cases are referred to as AMBIGUOUS, and the probability distribution $P_{\text{AMBIGUOUS}}$ is derived by applying the simplification from Equation (5.9) to $\text{P}_{\text{A-RHS}}$.

Let LHS $\rightarrow$ RHS be a rule in the form of Rule (5.4). Then we define $P(\text{RHS}|\text{LHS})$ as:

$$\begin{aligned} P_{\text{ALL}}(\text{RHS}|\text{LHS}) \;=\; & P(\text{UNAMBIGUOUS}) \cdot P_{\text{UNAMBIGUOUS}}(\text{RHS}|\text{LHS}) \\ & + P(\text{A-LHS}) \cdot P_{\text{A-LHS}}(\text{RHS}|\text{LHS}) \\ & + P(\text{A-RHS}) \cdot P_{\text{A-RHS}}(\text{RHS}|\text{LHS}) \\ & + P(\text{AMBIGUOUS}) \cdot P_{\text{AMBIGUOUS}}(\text{RHS}|\text{LHS}) \end{aligned} \tag{5.10}$$

We developed a method to pre-compute $P_{\text{ALL}}(\text{RHS}|\text{LHS})$ directly while training, without even having to explicitly compute the distributions on the right side of Equation (5.10). P_{ALL} fully specifies the probability distribution required to train a parser on the morphologically-tagged corpus. Having defined the probability distribution, we now turn to evaluating the effect of morphological features on parsing performance.

5.2.3 Method

We use the normal divisions of training, development and test sets. All three sets are tagged automatically, but only the training set contains the hand-annotated subset. When hand-annotated morphological tags are available, they are chosen over the automatically tagged ones. The morphological tags are propogated up the trees as described in Section 5.1.3. If the morphological tags are ambiguous, each data point in the training data is split and weighted as described in Section 5.2.2. Although the propogation, splitting and weighting worked successfully, we found the splitting and weighting step to be prohibitively time consuming. To make this step more efficient, we limited propogation inside a few key nodes -- NP, AP and PP, as well as their co-ordinated versions CNP, CAP and CPP. Notably missing from this set is the proper noun category MPN. We found that morphological taggers simply proposed too many tags for these categories, greatly slowing down the splitting and weighting step. Moreover, many of these tags cannot be pruned because of a sparsity of marked pruning cues such as articles. The problem was worse when MPNs were co-ordinated: we cannot guarantee that co-ordinate sisters share the same gender or number. Therefore, not only did we choose not to propgate the morphological tags of proper nouns, but they were stripped from the training set entirely.

In terms of evaluation, we continue to use the standard measures we have been using to this point. As all the hand-annotated morphological tags are in the training data, we do not evaluate the accuracy of labelling the morphological tags themselves on the training data. Rather, we test the effect the tags have on labelled bracketing.

We test four models, two with GFs and two without. The baseline is the parser from Chapter 4 which includes beam search and Brants smoothing, but no multipass parsing. The second model adds morphological tags to this parser. These two parsers make the first pair. The second pair of models includes GFs, including all the re-annotations proposed in Chapter 4. The first of this pair is the same as the best performing parser of Chapter 4. Again, the second includes the morphological tags.

5.2.4 Results

The results are shown in Table 5.5. The F-score of the morphological parser without GFs (Baseline+Morph) does not surpass the F-score of the baseline. However, it does acheive a slightly higher recall. Including GFs in the morph model was not successful. Comparing the parser with morphological tags and GFs (Baseline+Morph+GF) to the parser only including GFs (Baseline+GF), we find that Baseline+Morph+GF performs worse on every measure.

5.2.5 Discussion

The inclusion of morphological tags does not appear successful. Nevertheless, the small increase in recall does suggest that the morphological tags are pruning away some linguistically unlikely edges, although this effect is overshadowed by the effect of grammatical functions. There are a number of possible reasons why including morphological tags does not help.

First, there is still some noise in the training data. As we saw in Section 5.1, the morphological tags are only 91% correct on a word-by-word basis. We did not test how accurate the tags are when considered on a constituent-by-constituent basis. Too many mistakes in the training data would make it difficult to label words and constituents correctly while parsing.

Second, the tag set used was a ‘lowest common denominator’ choice, including only the attributes available in all of the morphological taggers and in the NEGRA corpus. Even using this diminished set required intensive translations between the theoretical and practical differences of the various tag sets. However, in doing so, some potentially useful tags were left out. In particular, the declension of adjectives changes based upon the pronoun preceeding them, yet it is not possible to describe these changes with the current set of attributes.

A third problem may be with the way the probabilities of the grammar model were estimated. Lacking correct training data, we relied upon an ad-hoc estimation. The first three problems all stem from a lack of suitable training data. Corpora including such data would obviously not have these problems.

A fourth problem is that there may be little room to improve the accuracy of POS tagging. Recall that some POS tag ambiguities faced by parsers or finite-state taggers are essentially resolved if better morphological information is present. A relevant example is adjective/verb ambiguity. While not very common, it can lead to disasterous errors while parsing. When an adjective is mistaken for a verb, it is often the case that the falsely tagged adjective has the wrong case, number or gender, or that the falsely tagged verb has the wrong tense or person. Indeed, solving this kind of ambiguity (common in co-ordinated structures) is one of the main justifications for including morphological tags. But if POS tagging is accurate enough, including morphological tags may not ‘buy’ much improvement. We will return to a closer evaluation of POS tagging accuracy in Section 6.1.

	Precision	Recall	F-score	Avg CB	0CB	$\leqslant$2CB
Baseline	74.4	70.8	72.6	0.66	66.2	91.7
Baseline+Morph	73.1	71.1	72.1	0.69	66.1	91.0
Baseline+GF	75.9	76.6	76.3	0.54	70.1	94.2
Baseline+Morph+GF	73.6	75.1	74.4	0.58	68.3	93.8

Table 5.5. Parsing with morphological features

A final problem may be that, despite the inclusion of smoothing, the parser may be facing sparse data problems. While there are not nearly as many states as in lexicalized parsing, each node is now getting quite complex. In Section 5.3, we develop an approach for overcoming sparse data by decomposing nodes into their constituent features, giving some indication if this the source of our difficulties.

5.3 Parsing with Attributes

If the failure of the morphology model is indeed due to sparse data, we may appeal to two of the standard approaches to overcoming overfitting: smoothing or making further independence assumptions. It is not immediately obvious how to introduce more smoothing in to the model. It is possible, though, to assume greater indepedence between grammatical categories and attributes. Doing so, the underlying grammar begins to resemble attribute-value grammars (Shieber, 1986; Johnson, 1988). Essentially, we end up describing a probability distribution over very simple attribute-value structures. Keeping attributes separate (or at least partially separate) from grammatical categories allows us to develop alternative estimates of the probability of the rule, by making different assumptions about how to assign probabilities to attributes such as OA or HD.

5.3.1 Parameterization

Stolcke (1994) first discussed the benefits of using maximum entropy models to assign probaiblity distributions over attribute-value grammars. A more detailed and forceful argument in favour of a related approach, log linear models, was due to Abney (1997). Maximum entropy and log-linear models are the prevalent approach to parameterizing attribute-value grammars. However, despite the popularity of such models, it is by no means true that these are the only approach to parameterizing attribute-value grammars. It is worth reviewing the argument of Abney (1997) to show how there are other possible solutions to the problem Abney presents.

Consider the following grammar:

$$\begin{bmatrix} \text{S} \\ \end{bmatrix} \rightarrow \begin{bmatrix} \text{A} \\ \boxed{1} \end{bmatrix} \begin{bmatrix} \text{A} \\ \boxed{1} \end{bmatrix}$$

$$\begin{bmatrix} \text{S} \\ \end{bmatrix} \rightarrow \begin{bmatrix} \text{B} \\ \end{bmatrix}$$

$$\begin{bmatrix} \text{A} \\ a \end{bmatrix} \rightarrow a$$

$$\begin{bmatrix} \text{A} \\ b \end{bmatrix} \rightarrow b$$

$$\begin{bmatrix} \text{B} \\ \end{bmatrix} \rightarrow a\,a$$

$$\begin{bmatrix} \text{B} \\ \end{bmatrix} \rightarrow b\,b$$

Up to notational changes, it is the same grammar used by Abney. In this grammar, the variable $\boxed{1}$ can take the values a or b. Now, consider the following tree:

$$\begin{array}{c} \begin{bmatrix}\text{S}\\ \end{bmatrix} \\ \diagup \quad \diagdown \\ \begin{bmatrix}\text{A}\\ a\end{bmatrix} \; \begin{bmatrix}\text{A}\\ a\end{bmatrix} \\ | \qquad\quad | \\ a \qquad\quad a \end{array}$$

To assign a probability to this tree, Abney, following Eisele, suggests the following 'incorrect' estimate:

$$\begin{aligned} & P\left(\begin{array}{c} \begin{bmatrix}\text{S}\\ \end{bmatrix} \\ \diagup \quad \diagdown \\ \begin{bmatrix}\text{A}\\ a\end{bmatrix} \; \begin{bmatrix}\text{A}\\ a\end{bmatrix} \\ | \qquad\quad | \\ a \qquad\quad a \end{array}\right) \\ = \; & P\left(\begin{array}{c} \begin{bmatrix}\text{S}\\ \end{bmatrix} \\ \diagup \quad \diagdown \\ \begin{bmatrix}\text{A}\\ \boxed{1}\end{bmatrix} \; \begin{bmatrix}\text{A}\\ \boxed{1}\end{bmatrix} \end{array} \middle| \begin{bmatrix}\text{S}\\ \end{bmatrix}\right) \cdot P\left(\begin{array}{c} \begin{bmatrix}\text{A}\\ \boxed{1}\end{bmatrix} \\ | \\ a \end{array}, \boxed{1} = a \middle| \begin{bmatrix}\text{A}\\ \boxed{1}\end{bmatrix}\right)^2 \end{aligned} \tag{5.11}$$

At this point, the probability assigned to the parse becomes troublesome. Abney provides example probabilities for the grammar, showing that the probability for this tree is too low. As Abney writes, "something has gone very wrong". However, our more explicit notation allows us to see the exact reason Abney and Eisele ran into difficulties: the variable substitution $\boxed{1} = a$ occurs twice, each time a is generated. Because the variable is substituted twice, it may be substituted with two different values. The example grammar Abney provides, however, allows only one substitution.

Abney proposes maximum entropy models as the solution to this double substitution problem. Recall that while derivations in context-free grammars are trees, derivations in attribute-value grammars (in general) are directed acyclic graphs (DAGs), which allow directed edges to rejoin. Noting that PCFGs can be seen as maximum entropy models where rules are features, Abney suggests using these rejoining subtrees as features, instead. While PCFGs have an efficient maximum likelihod estimate, the more complicated maximum entropy models proposed by Abney require a much more intensive training algorithm.

The notation we use suggests another possible solution: ensure the variable is assigned only once. There are two ways to go about this: one approach is to "split the state space", and instantiate all variables with values (essentially what we have been doing until this point). The second is to leave the variables uninstantiated, but only generate the assignment once. Compare the incorrect derivation in Equation 5.11 with the following:

$$P\left(\begin{array}{c} \begin{bmatrix}\text{S}\\ \end{bmatrix} \\ \diagup \quad \diagdown \\ \begin{bmatrix}\text{A}\\ \boxed{1}\end{bmatrix} \; \begin{bmatrix}\text{A}\\ \boxed{1}\end{bmatrix} \end{array} \middle| \begin{bmatrix}\text{S}\\ \end{bmatrix}\right) \cdot P(\boxed{1} = a) \cdot P\left(\begin{array}{c} \begin{bmatrix}\text{A}\\ a\end{bmatrix} \\ | \\ a \end{array}, \middle| \begin{bmatrix}\text{A}\\ a\end{bmatrix}\right)^2 \tag{5.12}$$

One way to ensure a variable is only assigned once is to force a variable to appear only once in a child. This is the approach used by Stolcke (1994). In Stolcke's models, the 'target' of the assignment is always a variable of the parent, and the assigned values are either inherited from a child below, or they are synthesized at the current rule. In Rule 5.1, the GF category OA (accusative object) is inherited. Linguistically, the parent inherits this category the noun, but in our simple grammar, only the article is marked as being accusative. Therefore, in this particular example, the parent inherits the GF from the article. To illustrate how the probabilities are calculated, consider a simplification of Rule 5.1, where the only attribute is GF. The probability of this rule is:

$$
\begin{aligned}
& P\left(\begin{bmatrix}\text{NP} & \\ \textsc{gf} & \text{OA}\end{bmatrix} \rightarrow \begin{bmatrix}\text{ART} & \\ \textsc{gf} & \text{OA}\end{bmatrix}\begin{bmatrix}\text{NN} & \\ \textsc{gf} & \text{NK}\end{bmatrix} \middle\| \begin{bmatrix}\text{NP} & \\ \textsc{gf} & \text{OA}\end{bmatrix}\right) \\
= \quad & P\left(\begin{bmatrix}\text{NP} & \\ \textsc{gf} & \boxed{1}\end{bmatrix} \rightarrow \begin{bmatrix}\text{ART} & \\ \textsc{gf} & \boxed{2}\end{bmatrix}\begin{bmatrix}\text{NN} & \\ \textsc{gf} & \boxed{3}\end{bmatrix} \middle\| \begin{bmatrix}\text{NP} & \\ \textsc{gf} & \boxed{1}\end{bmatrix}\right) \\
\cdot \; & P\left(\boxed{3} = \text{NK} \;\middle|\; \begin{bmatrix}\text{NP} & \\ \textsc{gf} & \boxed{1}\end{bmatrix}\right) \cdot P\left(\boxed{1} = \boxed{2} \;\middle|\; \begin{bmatrix}\text{NP} & \\ \textsc{gf} & \boxed{1}\end{bmatrix}\right)
\end{aligned}
$$

The assignment $\boxed{2} = \text{OA}$ is made when the POS tag ART generates its lexical element. Stolcke notes this approach does not work if extended to several variables (what is commonly referred to as having *re-entrancies*). We essentially run in to the same problem as noted above. This makes it difficult to properly model the MORPH attribute: in a noun phrase, the morph attribute will be the same for almost all preterminal children. Stolcke suggests two approaches: (preceeding Abney) random fields and using a 'head-driven' approach. In the head-driven approach, one child is picked as the head, and the attributes of the other children are condition upon that of the head child. To better illustrate how probabilities are calculated using this approach, consider another simplification of Rule 5.1 which only includes the morph attributes. The probability would be:

$$
\begin{aligned}
& P\left(\begin{bmatrix}\text{NP} & \\ \textsc{morph} & \text{Masc.Akk.Sg}\end{bmatrix} \rightarrow \begin{bmatrix}\text{ART} & \\ \textsc{morph} & \text{Masc.Akk.Sg}\end{bmatrix}\begin{bmatrix}\text{NN} & \\ \textsc{morph} & \text{Masc.Akk.Sg}\end{bmatrix}\right) \\
= \; & P\left(\begin{bmatrix}\text{NP} & \\ \textsc{morph} & \boxed{1}\end{bmatrix} \rightarrow \begin{bmatrix}\text{ART} & \\ \textsc{morph} & \boxed{2}\end{bmatrix}\begin{bmatrix}\text{NN} & \\ \textsc{morph} & \boxed{3}\end{bmatrix} \middle\| \begin{bmatrix}\text{NP} & \\ \textsc{morph} & \boxed{1}\end{bmatrix}\right) \\
& \cdot P\left(\boxed{1} = \boxed{3} \;\middle|\; \begin{bmatrix}\text{NP} & \\ \textsc{morph} & \boxed{1}\end{bmatrix}\right)
\end{aligned}
\tag{5.13}
$$

An external constraint (with certain probability) enforces that $\boxed{2} = \boxed{3}$. The value of $\boxed{3}$ is set to Masc.Akk.Sg when the noun expands to a word.

A third approach, introduced by Schmid (2002), is to generate the assignment at the first moment it is needed. This ties the parameterization to a particular parsing model. In an incremental left-to-right parser, the assignment is generated when the left most child is generated. If we wanted to parse the string *den Mann* with Rule 5.1 (again only with morph attributes) the probability may be estimated as:

$$
\begin{aligned}
& P\left(\begin{bmatrix}\text{NP} & \\ \textsc{morph} & \text{Masc.Akk.Sg}\end{bmatrix} \rightarrow \begin{bmatrix}\text{ART} & \\ \textsc{morph} & \text{Masc.Akk.Sg}\end{bmatrix}\begin{bmatrix}\text{NN} & \\ \textsc{morph} & \text{Masc.Akk.Sg}\end{bmatrix}\right) \\
= \; & P\left(\begin{bmatrix}\text{NP} & \\ \textsc{morph} & \boxed{1}\end{bmatrix} \rightarrow \begin{bmatrix}\text{ART} & \\ \textsc{morph} & \boxed{1}\end{bmatrix}\begin{bmatrix}\text{NN} & \\ \textsc{morph} & \boxed{1}\end{bmatrix} \middle\| \begin{bmatrix}\text{NP} & \\ \textsc{morph} & \boxed{1}\end{bmatrix}\right)
\end{aligned}
$$

Then the probability of expanding ART into *den* would be:

$$
P\left(\begin{bmatrix}\text{ART} & \\ \textsc{morph} & \text{Masc.Akk.Sg}\end{bmatrix} \rightarrow \textit{den}, \boxed{1} = \text{Masc.Akk.Sg} \middle\| \begin{bmatrix}\text{ART} & \\ \textsc{morph} & \boxed{1}\end{bmatrix}\right)
$$

	Precision	Recall	F-score	Avg CB	0CB	$\leqslant$2CB
Baseline	74.4	70.8	72.6	0.66	66.2	91.7
Baseline+GF	75.9	76.6	76.3	0.53	70.1	94.2
Decompose GF	75.1	75.5	75.2	0.55	69.3	93.1
Decompose Morph	71.2	71.3	71.2	0.76	63.1	90.7
Decompose GF+Morph	72.2	74.2	73.2	0.65	64.8	92.2

Table 5.6. Parsing with node decomposition

The grammar also requires some probabilistic 'clean up' while training to remove probability mass from impossible derivations.

Notice that variables perform two different functions. First, variables replace instantiated values in context-free rules, allowing us to pool parameter estimates. Second, variable assignment by declarations of equality also replace instantiated values. The second use is most transparent in Stolcke's approach. There is an explicit statement in Equation 5.13 calculating the probability that $\boxed{1}=\boxed{2}$. This replaces potential instantiations. For example, if X_1 is the trace attribute of the parent and X_2 is the trace attribute of the child, then $P(\boxed{1}=\boxed{2})$ replaces $P(X_1=\text{Masc.Akk.Sg}, X_2=\text{Masc.Akk.Sg})$, $P(X_1=\text{Fem.Akk.Sg}, X_2=\text{Fem.Akk.Sg})$, etc. While we may worry that rule probabilities may overfit the training data, the tag set of morphological labels is quite small, making these probabilities relatively easy to estimate.

This suggests an additional approach to parameterizing attribute-value grammars: we may use variables in context-free rules but instantiate values for variable assignments. In essence, this means the probability of context-free rules can be computed as before (just like with Stolcke's method), but we add the additional step of computing attributes. To account for several children which may have the same value, we condition upon the attributes generated for previous sisters. Thus, we may calculate rule probabilities in the following way:

$$\begin{aligned}
& P\left(\begin{bmatrix} X_i & \\ \text{GF} & G_i \\ \text{MORPH} & M_i \end{bmatrix} \middle| \begin{bmatrix} X' & \\ \text{GF} & G' \\ \text{MORPH} & M' \end{bmatrix} \rightarrow \ldots \begin{bmatrix} X_{i-2} & \\ \text{GF} & G_{i-2} \\ \text{MORPH} & M_{i-2} \end{bmatrix} \begin{bmatrix} X_{i-1} & \\ \text{GF} & G_{i-1} \\ \text{MORPH} & M_{i-1} \end{bmatrix}\right) \\
= \; & P_{\text{CFG}} \cdot P_{\text{MORPH}} \cdot P_{\text{GF}}
\end{aligned}$$

Where P_{CFG} is defined as before:

$$P_{\text{CFG}} = P(X_i | X' \rightarrow \ldots X_{i-2}\, X_{i-1})$$

P_{GF} is defined as:

$$P_{\text{GF}} = P(G_i | X_i, X', M', M_{i-1}, M_{i-2})$$

And P_{MORPH} is defined similarly to P_{GF}.

This parameterization is not as ambitious as that of Abney or Schmid. Because the set of features we are using is quite simple, we only need to consider atomic attributes. Indeed, the attributes we use are quite simple largely due to one simplifying assumption: unlike formalisms which store entire parses in a single attribute-value matrix, our attribute-value structures are simply nodes on a parse tree. Of course, this assumption makes it harder to interpret parses: it is usually easier to read values from a single-attribute value structure rather than a tree. Theoretical complaints aside, having chosen a parameterization for attribute-value decomposition, we may now turn to testing the models.

5.3.2 Method

We run two sets of experiments. The first set tests the effect of node decomposition on GFs alone, the second tests the effect on morphological tags, as well.

Four models are tested. These are: a baseline without GF labels at all, a baseline with GFs, and two models with two differing approaches on how to decompose GF labels. In all models with GFs, we use the GF transformations of the best-performing model of Chapter 4. Each model is then tested using both no smoothing and with Brants smoothing.

5.3.3 Results

The results are summarized in Table 5.6. None of the models with decomposition were able to beat the model with undecomposed GFs (Baseline+GF). In addition, the model with decomposed attributes and morphological features (Decompose Morph) did not even beat the baseline without GFs, although once again the recall was higher. The morpholgical attributes continued to pose some problems as the model with decomposed GFs and morph features (Decompose GF+Morph) performed worse than the decomposition model with GFs alone (Decompose GF).

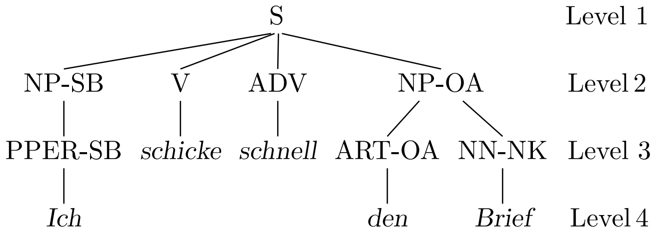
The decomposed models did have some successes, however. The Decompose GF model did acheive a higher F-score than the baseline without GFs (75.3 versus 72.6), although it performed slightly worse than the Baseline+GF's 76.3.

5.3.4 Discussion

This disappointing results can be mitigated, possibly, to two factors: the decomposition model is not appropriate to these attributes or there is either no sparse data or the decomposition model is not appropriate to solve the sparse data problem. We can say with some confidence that the decomposition model is not completely inappropriate: the Decompose GF model did outperform the baseline. This result suggets that GFs impart the parser with useful information even when decomposed from nodes. Nonetheless, it is possible to argue the decomposition model is not appropriate in some other ways.

Decomposing attributes from rules is tantamount to assuming the values of the attributes to not affect rule probabilities. To illustrate this, consider the nonterminals NP-SB, NP-OA and NP-DA. By decoupling the features (SB, OA and DA) from the nonterminal, we assume that (i) any NP can appear in the same place in the sentence (because rules like S $\rightarrow$ NP-SB V NP-OA are now just S $\rightarrow$ NP V NP), and (ii) that all NP's expand the same way (because rules like NP-SB $\rightarrow$ PPER-SB and NP-OA $\rightarrow$ ART-OA NN-NK become just NP $\rightarrow$ PPER and NP $\rightarrow$ ART NK).

This is not always the case. For example, while it may make sense to encode case as a variable, we loose the ability to model several things. First, nominative NPs appear in different parts of the sentence as dative or accusative NPs; this information would be lost by not instantiating the variable. Second, nominative NPs are more likely to expand to pronouns. Finally, each may be lexicalized differently, especially in the articles and pronouns. Each problem is depicted in the figure below:



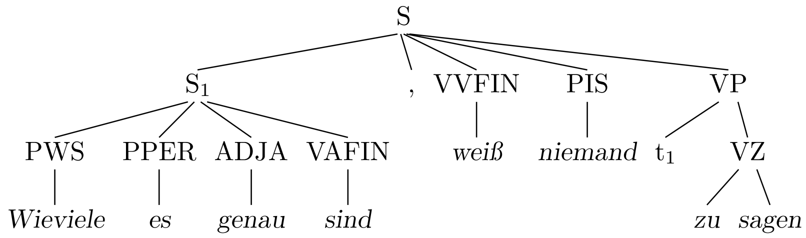
On Levels 1 and 2, we have information about where the NP-SB and NP-OA constituents may appear. On Levels 2 and 3, we have information about how nominative and accusative NPs expand: the nominative is more likely to be a pronoun than the accusative. Finally, on Levels 3 and 4, the POS tags expand to words. Decomposing GFs from the syntactic categories may well have alleviated sparse data problems, but this problem appears to be dominated by the effect of GF labels on the probabilistic behavious of syntactic categories. We can similarly discount sparse data as a cause of the results of Section 5.5 above.

5.4 Gap Features

Let us leave morphology aside for a moment, and turn to another aspect of German syntax. As we have already noted, NEGRA's flat annotation was developed because of German's semi-free word order. Unfortunately, some freer word order constructions cannot be sufficiently analyzed using flat annotations alone. Consider the following sentence:

	Wieviele	*es*	*genau*	*sind*	,	*weiß*	*niemand*	*zu*	*sagen*
Example 5.6.	How many	they	precisely	are		knows	no one	to	say
	"No one can say exactly how many of them there are"								

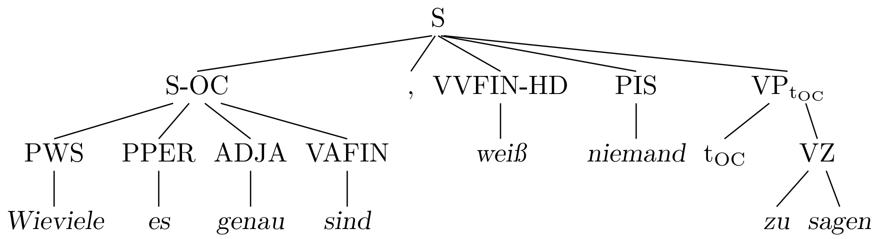
The clause "*Wieviele es genau sind*" has been fronted. If it had been a dependent of the main verb, it would simply be given an OC (clausal object) GF. Here, however, it is a long-distance dependent of *sagen*, which is inside a VP. In the Penn Treebank version of NEGRA, this is annotated by the *empty element* t_1 in the VP, which is co-indexed (by the numeral '1') with its antecedent S_1:[5.1]



Other topicalized constructions are also analyzed as LDDs. Among these are *partially fronted* VPs, where the nonfinite verb of a VP may be fronted, leaving some or all of its dependants behind. In addition to topicalization, another common case where NEGRA posits LDDs is extraposition, when a dependent appears after the VP.

A proper treatment of LDDs is important: such dependencies occur in 27% of all sentences in our development section. Clearly, these sentences cannot be fully analyzed without paying attention to LDDs. There are also other benefits to accounting for LDDs: Dienes and Dubey (2003b) show that an unlexicalized parser which includes a model for LDDs is better at finding local dependencies than one which ignores LDDs.

While there are many approaches to handling LDDs (see Dienes, 2004) for a full discussion), the models of Dienes and Dubey (2003a) and Dienes and Dubey (2003b) are particularily interesting: they allow the use of PCFG models, they are fairly simple, and perform well. The basic approach of Dienes and Dubey, following Gazdar et al. (1985), is to thread a path between the empty element (EE) and the antecedent. Such *gap threading* works picking a common label for the EE and antecedent, and marking that label on each constituent dominated by the EE which is not also dominated by the antecedent. These labels are referred to as `gap+` attributes. For our experiments, the label is the GF of the antecedent. To illustrate this, we annotate the tree above with the `gap+` attribute t_{OC}:



5.1. Normally, the sentential complement would be extraposed after *zu sagen*. However, this positioning would itself involve a long-distance dependency, hence the position of the trace before *zu sagen*.

5.4.1 Method

We use the same training, development and test sections as before, but all trees with LDDs are modified to include threading of gap+ attributes. Empty elements are treated as additional words. As EEs do not appear in normal text, a realistic parser ought to insert them before or during parsing. We do not consider this problem here. Instead, we insert EEs wherever they appear in the gold standard. This may artificially inflate results in two ways. First, finding the site of antecedents is relatively difficult in English (Dienes, 2004), and one might expect it to be harder in German. Second, because the gap+ features contain the grammatical function of their antecedent, the parser may know to expect a constituent with such a GF. Nonetheless, this approach has the advantage that it is simple and is capable of illustrating the importance of LDDs.

We report results in three conditions: the baseline, a parser with traces and a parser with traces and gap threading. The baseline is the best GF parser from Chapter 4. The baseline parser is not given empty elements in the input nor does it include gap+ features on node labels. The second parser is given empty elements in the input and includes trace features on the antecedent. The third parser is an extension of the second which includes gap threading in the form of gap+ features.

5.4.2 Results

The results are summarized in Table 5.7. The parser which includes empty elements (EE) performs worse than the baseline on every metric we measure. Most notably, F-score falls from 76.3 to 73.9. In addition, there are more crossing brackets on average. When gap threading is included, however, the situation is different. The parser with gap threading (EE+Thread) performs better on labelled bracketing, with an overall F-score of 77.4. The crossing bracket measures are also higher, with an average crossing bracket of 0.54.

5.4.3 Discussion

Giving a treatment to LDDs is beneficial even for finding local dependencies, replicating the finding of Dienes and Dubey (2003b) for German. However, this is only true after re-annotation: threading gap+ categories. The need to perform a re-annotation to improve results parallels the finding in Chapter 4, with re-annotated GFs.

There are some parts missing from LDD model. First, we have no mechanism for automatically finding the site of EEs. This is left for future research. Second, we have not measured how well the parser finds LDDs themselves, or on constructions which often depend upon LDDs. To the extent these two can be separated, we return to the former in Chapter 6, sufficing to investigate the latter in Section 5.5.

5.5 Traces and Verb Final Clauses

In Section 3.6 and Section 4.4, we tested the efficacy of the sister-head and GF parser on verb-final and topicalization constructions. We found the parsers had difficulty with both types of constructions. Noting that sentences containing these constructions tended to be longer, and that longer sentences are in general harder to parse, we then introduced a re-weighting scheme to account for the effects of sentence length. In both cases, we found that accounting for sentence length did not change the final results: these constructions are difficult to parse for the sister-head and GF models.

	Precision	Recall	F-score	Avg CB	0CB	$\leqslant$2CB
Baseline	75.9	76.6	76.3	0.53	70.0	94.2
EE	73.8	74.0	73.9	0.78	65.6	90.6
EE+Thread	77.2	77.7	77.4	0.54	70.3	94.1

Table 5.7. Parsing with long-distance dependencies

		ALL	VF	NOVF	TOPIC	NOTOPIC
Avg. Sentence length		7.5	11.2	6.4	8.9	6.5
Standard F-score	Baseline	76.3	73.2	77.8	75.6	76.9
	Thread	77.5	74.6	78.8	76.7	78.0
Weighted F-score	Baseline	76.3	75.2	76.8	76.0	76.2
	Thread	77.5	77.5	76.9	77.4	77.6

Table 5.8. Performance on various syntactic constructions

It is possible there is another confound: the presence of long-distance dependencies. On the development set, 51% of sentences with a verb-final clause and 45% of sentences with a topicalization construction also contain some kind of long-distance dependency. In contrast, only 20% of sentences without verb-final clauses and 14% of sentences without fronting contain a long-distance dependency. As we have now introduced a model which can handle long-distance dependencies, it is reasonable to suggest that it may remove this confound.

To test this hypothesis, we examine the performance of the gap-threading parser from Section 5.4 against a baseline which does not model LDDs (once again, the best-performing GF parser from Chapter 4 serves this purpose). As in Section 4.4, we alternately partition the data in two ways: sentences which contain some verb-final clause vs. those which do not, and sentence which contain some topicalized clause vs. those which do not. With each partition, we report results using F-scores of labelled bracketing and weighted F-scores, which attempt to accounts for sentence length effects (see Section 3.6 for a further discusion of weighting).

5.5.1 Results

The results are in Table 5.8. Including traces and threading leads to improvements in all conditions, measuring with both standard and weighted F-scores. Using standard F-scores, trace threading lead to a 1.4 point improvement in sentences containing a verb-final construction (VF), and a 1 point improvement in sentences which did not (NOVF). Furthermore, there was a 1.1 improvement in both sentences containing a topicalization construction (TOPIC) as well as those without (NOTOPIC).

When considering weighted F-scores, the relative increase in the VF case was much higher. The improvement for VF was 2.9 points, to a total of 77.5. This is actually higher than the 76.9 reported in the NOVF condition, iself a 0.1 point increase over the baseline. Both the TOPIC and NOTOPIC cases saw an improvement of 1.4 over the baseline, for final score of 77.4 and 77.6 respectively.

5.5.2 Discussion

Introducing gap threading made it easier to recover verb-final constructions. Indeed, the weighted result in sentences with verb-final constructions was a bit higher than in those sentences without such constructions. It would be comforting to assume this is because VF sentences are actually easier to parse after accounting for the effects of sentence length. There are, however, two other possible causes of this result. First, the fact we were using perfect traces may have given an unrelatistic boost to the trace model. Second, the weighting scheme may be overeager, and give too much weighting to smaller sentences. It is possible that all three may influence the result, although it is difficult to determine to what extent.

In constrast to the success on verb-final sentences, the improvement in the TOPIC and NOTOPIC conditions were more balanced. The improvement due to gap threading was the same in both conditions, meaning that, for our model, parsing topicalized sentences does not benefit from the inclusion of empty elements and gap threading. This is true despite that fact that long-distance dependencies are actually more common in the TOPIC condition than the VF condition. A possible explanation might be the nature of the long-distance dependencies present in each condition. The baseline parser has difficulty with relative clauses, especially extraposed relative clauses -- it often mistakes relative pronouns for articles, and attempts to attach the verbs elsewhere in the sentence, leading to many attachment errors. The parser with gap threading does better in these situations, which helps in the VF condition more than in the TOPIC condition.

Moreover, long-distance dependencies specifically involving topicalization are usually not as hard to parse. The most common fronted constituents are modifiers, which do not cause any special difficulties when missing from the *Mittelfeld*. Troublesome constructions, like partial VP fronting, are extremely rare in the NEGRA corpus.

5.6 Conclusions

The fundamental purpose of this chapter was to use a PCFG to model syntactic properties of German which are often not part of parsing models in English, in particular morphology and long-distance dependencies. The results were mixed. While the parsers with morphological attributes had some moderate successes, the overall effect was a less accurate parser. The outcome of modelling non-local dependencies, however, was more favorable. When including gap threading, the parser even did a better job of finding local dependencies.

It is highly probable that a cause of the difference may be the way the data was generated: the morphological attributes were almost entirely machine-generated, whereas the non-local depdendencies were almost entirely annotated by humans. If anything, this may suggest that the opposing results with morphological vs. non-local attributes was not because morphology is not useful while parsing, but because human annotators tend to be more accurate than morphological taggers.

Chapter 6
Further Evaluation

Throughout this thesis, we have been evaluating the accuracy of parsing models with labelled bracketing scores and, to a lesser extent, consistent bracket scores. These measures alone do not tell us everything we wish to know about the accuracy of a parser. For instance, a correct semantic interpretation depends on accurate POS tags and GF labels as much as it depends on accurate parsing. Despite this, we have not yet measured how accurate any parser is at applying POS tags or GF labels.

Moreover, it is not clear that labelled bracketing scores are necessarily the best way to measure the accuracy of trees. Lin (1995) argues that labelled bracketing scores are susceptible to *cascading errors*, where one incorrect attachment decision will cause the scoring algorithm to count more than one error. Lin suggests using word-word dependencies as the basis of an alternative evaluation metric which is not prone to the same problems. Dependency measures count the percentage of head-dependant relationships the parser correctly finds.

Dependencies are important for others reasons. Hockenmaier (2003) argues that dependencies are more annotation neutral than labelled bracketing scores. She finds her binary-branching grammar performs poorly using labelled bracketing scores, but nonetheless has comparable performance to Collins (1997) when using unlabelled dependencies.

In this chapter, we present further evaluation of the models in previous chapters with the goal of assessing their success at finding correct POS tags (Section 6.1), GF labels (Section 6.2) and word-word dependencies (Section 6.3). We concentrate the evaluation on baselines and the best-performing models of each chapter: the unlexicalized baseline, the sister-head parser, an unlexicalized smoothed baseline, the best smoothed GF parser and the gap-threading parser.

6.1 POS Tagging

Testing the accuracy of POS tagging is interesting for three reasons. First, correctly guessing POS tags is an important goal in and of itself. Second, because POS tagging has developed into a field of its own right, it is reasonable to ask if broad-coverage parsers render stand-along POS taggers obsolete. Although stand-alone taggers are simple to build, and tag both quickly and accurately, if parsing is accurate and fast enough we may question the importance of tagging as an independent step. Finally, an incorrect POS tags can have negative ramifications on further parsing decisions. It is worth investigating how often and to what degree a wrong POS tag can mislead a parser.

We measure the accuracy of POS tagging by simply comparing how often the guessed tag matches the gold standard. Because there is one and only one guess per word, we need not measure precision and recall. Four models are tested: the unlexicalized baseline, the baseline with grammatical functions, the model with smoothing but without grammatical functions, and, finally the model with smoothing and grammatical functions. Note that the unlexicalized baseline is more than a simple PCFG: it includes Markovization and suffix analysis for guessing POS tags of unknown words. The POS tagging accuracy of the sister-head parser is not comparable to the other parsers here, and hence it is not included in the evaluation. While the sister-head parser does guess the POS tag of many words, for unknown words, it requires an annotated POS tag from a corpus.

Baseline	96.5
Baseline+GF	96.4
Smooth	96.4
Smooth+GF	96.7

Table 6.1. POS tagging accuracy

	Precision	Recall	F-score	Avg CB	0CB	⩽2CB
Smooth+GF	75.9	76.6	76.3	0.53	70.1	94.1
Smooth+GF+Perfect Tags	85.4	85.0	85.2	0.27	82.7	98.1

Table 6.2. Results with perfect tagging

In addition to evaluating POS tagging alone, we also re-ran the best-performing parser (the smoothed model from Chapter 4) with perfect tags. In other words, the parser used the correct POS tags from the gold standard trees instead of guessing the tags itself, just like the parsers in Chapter 3. Using perfect POS tags allows us to gauge the impact of POS tagging errors on parsing mistakes.

6.1.1 Results

Table 6.1 shows the accuracy of POS tagging, and Table 6.2 has the results of parsing with perfect tags. Looking at the POS tagging results first, the baseline model had a score of 96.5. The Baseline+GF and Baseline+Smooth models were both slightly lower at 96.4 each. The Smooth+GF model produced the highest result of 96.7. Note that both models with GFs ingore the GFs during evaluation.

Giving the Smooth+GF model the correct POS tags improves the labelled bracketing F-score of the Smooth+GF model from 76.3 to 85.2. The average crossing bracket falls to 0.27 compared to 0.53, the 0CB figure rises to 82.7% of sentences from 70.1%, and the percentage of sentences with less than two crossing brackets rises to 98.1 from 93.2.

6.1.2 Discussion

The best result in NEGRA POS tagging known to us is 96.7 by the TnT tagger reported by Brants (2000). The best result here (by the Smooth+GF model) also gives a score of 96.7 with the same amount of training data. Strictly speaking, though, the result of the Smooth+GF model is not comparable to the results with TnT because Brants makes use of multifold cross-validation testing and we do not. After re-trained and re-tested TnT using our split of training and testing data, the tagger still achieves an accuracy of 96.7. Therefore, we may conclude that the best parsing model here matches the best tagging results in German.

Despite the closeness of the results, the Smooth+GF parser is considerably more complex than Brants' POS tagger in at least three ways: first, it is a parser rather than a finite-state tagger; second, due to the use of GFs it has many more possible states, even in POS tags; and third, it makes much more intensive use of smoothing. The last point is particularly relevant: due to time taken to calculate smoothed probabilities, the Smooth+GF parser is much slower than other parsers, let alone a tagger.

The second experiment shows that accurate POS tagging is fundamental to accurate parsing. Giving the parser the correct POS tags raises labelled bracketing scores by nearly 10% and makes the average crossing brackets figure fall by nearly half. To determine how POS tags influence parsing errors, we performed an error analysis on 100 sentences in the development section. All of these sentences contained at least one POS tagging error.

Error Type	Frequency
Common/proper noun	35
Adjective/adverb	4
Interrogative/relative pronoun	3
Others	20
All errors	62

Table 6.3. *Lexical* POS Tagging Errors (see Section 6.1.2.1)

Error Type	Frequency
Adjective/verb	14
Improper verb tense	9
Common/proper noun	6
Errors with *als* ('as')	5
Preposition/adverb	4
Conjunct mistagging	3
Pronoun/article	3
Other closed-class words	15
Other open-class words	13
All errors	72

Table 6.4. *Structural* POS tagging errors (see Section 6.1.2.1)

6.1.2.1 Lexical and Structural Part-of-Speech Tagging Errors

Tagging errors come in two varieties: in one case, which we will refer to as *lexical* errors, the POS tagging mistake does not cause any obvious parsing mistakes. In the second case, which we call *structural* errors, a tagging mistake directly causes a parsing mistake.

About half the sentences we examined only contained lexical errors. We further subcategorized lexical errors by the type of POS tags which were confused. The result of this analysis is shown in Table 6.3. The first column shows the type of error, and the second shows the frequency of the error in the set of sentences we examined. The errors are ranked by decreasing frequency. The remaining error analyses in this section also use the same table layout. Although all these tables are ranked by decreasing frequency, there may be some variance in the ranking due to sampling. Because the error analyses are merely illustrative, we do not perform any significance tests to determine the effect of sample variance on the rank order. Indeed, errors which differ by only several occurrences will probably be ranked differently when given a different random set of 100 sentences.

Despite the possible problems due to sampling, it is clear from Table 6.3 that the most common lexical error by far is to confuse common nouns and proper nouns. Unlike English, capitalization does not provide any clues to distinguish between the two: both proper nouns and common nouns are capitalized. Both usually head an NP, so mistakes are usually localized.

While less common, other noticeable lexical errors are ambiguities between predicative adjectives and adverbs (both have the same lexical form), and between interrogative and relative pronouns (words like *was*, "what/which" may be used in both situations). Other lexical errors were too rare to be grouped into meaningful categories, although many of them involved either adjectives or adverbs.

Adjectives and adverbs were also at the root of many structural errors. The most common structural error was to mistake an adjective for a verb or vice versa. A complete list of the most frequent structural POS tagging errors can be seen in Table 6.4. The second most common mistake after adjective/verb errors were problems with verb tense, i.e. mistaking a finite verb for an infinitive. Both errors are essentially problems due to a lack of morphological features. This confirms the intuition from Chapter 5 that morphological attributes ought to be useful for parsing, although it appears that information about verb tense may be as important or more important than information about case, number and gender.

Error Type	Frequency
PP Attachment	21
Adverb Error	12
Co-ordination Errors	10
VP Too Small	6
Subordinate/Relative Clause Attachment	5
Extra/Missing Unary Node	4
Mistaking Main Verb/Subordinate Clause	3
Chunking Error	3
Others	6
Total	70

Table 6.5. Parsing errors with perfect tags

Although information about verb tense might help, it is also possible that some ambiguities may be resolved by simply using a larger training set. Most tagging errors are due to unseen words (Brants, 2000). Some parsing errors, however, are due to closed-class words which are ambiguous between two or more POS categories. *Als* ('as') may be used as a preposition, as a conjunct or for comparisons. In general, these can be difficult to tell apart without semantic information. However, *als* is only used as a conjunct together with *sowohl*. Either adding attributes or a careful lexicalization could distinguish the use as a conjunct with the other uses. Several other words are ambiguous between a use as an adverb and a use as a closed-class word. *Bis* ('until') and *über* ('over/via') are normally prepositions, but may act as adverbs, just as *aber* is normally a conjunct, but may be used as an adverb. Mistagging these words leads to severe parsing errors: if the parser believes it must create a prepositional phrase or several co-ordinated constituents, it will attempt to do so.

Overall, structural POS tagging errors have a profound impact on parsing accuracy. The average F-score of sentences with a lexical error is 80.2 versus 65.1 for sentences with a structural POS tagging error. Unlike the analysis of verb-final constructions of Chapter 3, it is safe to compare these two numbers: the average length of sentences with lexical errors is 15.6 words versus 15.3 words for sentences with structural errors. Indeed, if we take the same set of sentences, and re-parse them with perfect tags, the average F-scores are 88.3 for those which originally had structural errors and 88.2 for those which originally had lexical errors.

Using perfect tags does improve the F-score of sentences which only constitute lexical errors. This improvement seems paradoxical given the definition of what constituents a lexical error (recall that lexical errors are POS tagging errors which do not appear to cause parsing errors). The apparent paradox can be explained by a rather technical decision: we do not consider an incorrect GF label to constitute a POS tagging error. In other words, sentences which only contain lexical POS tagging errors may contain structural GF tagging errors. These errors are fixed when we re-parsed these sentences with perfect tags.

6.1.2.2 Parsing Errors not due to Part-of-Speech Tags

As noted above, giving the parser perfect POS tags does not solve all parsing errors. Many forms of attachment errors remain. We again performed a detailed error analysis on 100 sentences, using a different set of sentences than those use to classify POS tagging errors. We show the results in Table 6.5. The first column shows the attachment type, and the second column shows the frequency of the error in the set of sentences we examined. The most frequent problem is the attachment of modifiers such as PPs, subordinate clauses and NP appositions. Among these constituents, PPs are by far the most common, with subordinate clauses and other modifiers well behind.

The second most common error is improper attachment of adverbs. While adverb attachment errors could be grouped with the more general category of modifier attachment problems, an idiosyncrasy in NEGRA obliges us to treat adverbs separately. Linguistically, adverbs modify verbs or prepositions. However, in NEGRA, while some adverbs (such as *jetzt* 'now') do indeed modify verbs, others (like *auch* 'also/too') usually modify nouns. The parser often confuses the two cases. Better classification of the adverbs or lexicalizing the grammar could help reduce the impact of this problem.

Not all adverb attachment problems are due to annotation mistakes. If an adverb precedes a verb-modifying preposition, there is a genuine ambiguity between the adverb modifying the preposition or the verb. However, this class of adverb attachment mistakes is not nearly as common as the NP/verb attachment ambiguities noted above.

Problems with co-ordination are about as frequent as adverb attachment. There is not one single cause of co-ordination problems. At times, the parser may co-ordinate the wrong type of constituent, i.e. posit co-ordinated Ss rather than co-ordinated NPs. Some co-ordination mistakes are due to finding the wrong number of co-ordinated sisters. Other errors arise because commas are used with constructions other than co-ordination. For example, the parser may mistakenly believe that a comma indicates the presence of an extraposed constituent rather than a co-ordinate sister.

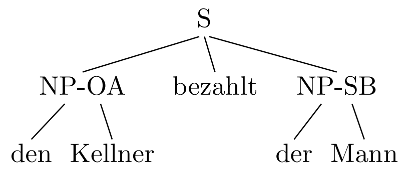
Finding the wrong number of co-ordinate sisters may be related to another problem: finding the correct boundaries of VPs. Recall that finding VP boundaries was also a problem for the lexicalized parsers of Chapter 3. In all cases where there was a problem with VP boundaries, the VP was too small. It is difficult to say why this happens, although Markovization may play a role. When a VP is too small, the parser is essentially choosing to attach a dependent to the auxiliary rather than the main verb. For many constituents, the probability of such an attachment ought to be zero, but it can be higher if the parser 'forgets' about the auxiliary. This may be solved by adding an attribute to denote the verb type, by using a longer Markov history, or forgoing Markovization altogether.

A related problem are cases with composed verbs where a parser 'forgets' there must be a VP, and instead declares that the main verb of the main clause is actually the head of a subordinate clause. Another common error is either including an extra unary production, or leaving a unary production out. The final parsing problem is chunking errors, i.e. the parser found the wrong boundary of a constituent covering POS tags.

6.2 Grammatical Functions

In Chapter 4, we saw that including grammatical functions in a parser dramatically improved parsing accuracy. However, we did not test the accuracy of the GF labels themselves. That is a large oversight: GF labels can be used to indicate the subject and objects of a verb. Because of semi-free word ordering, we cannot reliably expect the subject to be sentence-initial. Hence GFs labels are probably the easiest way to find the subject of a sentence -- a requirement for even the most basic syntactic analyses. Therefore, it would be insightful to test how accurately the parser applies these labels.

The simplest was to evaluate GF labels is to to treat the GF label as part of the node label. Consider the sentence '*den Kellner bezahlt der Mann.*' The correct parser of this sentence has an accusative NP spanning '*den Kellner*' and a nominative (subject) NP spanning '*der Mann*':



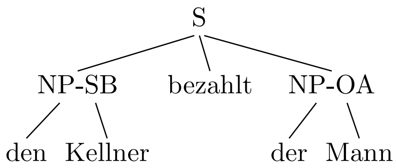
	Syn F-score		GF F-score		Syn+GF F-score	
	POS	Brackets	POS	Brackets	POS	Brackets
Baseline+GF	96.4	73.1	91.3	67.3	89.5	64.6
Beam+GF	96.3	72.6	91.7	64.8	89.8	62.6
Smooth+GF	96.7	76.3	91.8	67.8	90.0	65.7
Smooth+GF+Perfect Tags	--	85.2	--	77.5	--	76.2

Table 6.6. POS tagging and labelled bracketing results with grammatical functions

GF Type	GF Label	Baseline+GF	Beam+GF	Smooth+GF	Perfect Tags
Modifier	MO	56.1	55.1	58.6	67.2
Subject	SB	59.3	58.1	61.8	78.4
Clausal Object	OC	54.3	50.5	50.9	62.7
Postnominal Modifier	MNR	55.1	51.7	56.4	61.5
Accusative Object	OA	44.2	48.1	51.5	70.6
Postnominal Genitive	GR	76.3	76.2	78.7	86.4

Table 6.7. Labelled bracketing results by type of grammatical function

A possible incorrect parse might reverse the GF labels:



Using the evaluation measures used in Chapters 3, 4 and 5, the GF labels are always stripped, so both trees are considered to be correct. If the GF labels are not stripped, the first tree still has a precision and recall of 100%, but the 2$^{\text{nd}}$ only gets one node right out of three, resulting in a precision and recall of only 33%.

This is an unusually harsh metric. While there are only 29 node labels for syntactic categories, combining GFs with syntactic categories results in 291 labels -- an order of magnitude more. To make the comparisons more fair, we measure two different cases: brackets labelled with syntactic categories and GFs, and brackets labelled with GFs alone. In addition, because POS tags are also annotated with GF labels, we also measure the accuracy of these 'lexical' GF labels as well as lexical GF labels combined with POS tag labels.

6.2.1 Results

Table 6.6 summarizes the overall results. The columns in this table are grouped into pairs. The first of each pair measures results on POS tags, the second of each pair measures results on brackets. Each pair uses a different notion of labelling: either Syn (only the syntactic category must match the gold standard for a label to be correct), GF (only grammatical functions must match) or Syn+GF (both the syntactic category and the grammatical function must match). The 'Syn' columns simply re-state previous results for the sake of comparison.

The Baseline+GF model achieves a labelled bracketing F-score of 67.3 when brackets are labelled by GFs and 64.6 when brackets are labelled by GFs and syntactic categories. The corresponding figures for the Smooth+GF model are 67.8 in the GF condition and 65.7 in the Syn+GF condition.

Table 6.7 shows the individual results for the most common GF types. Each row in the table represents a different function type: modifier (MO), subject (SB), clausal object (OC), postnominal modifier (MNR), accusative object (OA) and postnominal genitive (GR). The GFs are listed by decreasing frequency, with the most common (MO) on top.

6.2.2 Discussion

Similar to the behaviour seen with syntactic categories, the Smooth+GF achieves higher POS tagging and labelled bracketing scores than the Baseline+GF or Beam+GF models.

Showing the results by GF type offers some insights to how various models effect GF labelling. The perfect tagging models is particularly useful to diagnose problems. Just as in Section 6.1, using correct POS tags improves overall results by about 10%. The effect within GF categories is quite different, however. Several GF categories, such as SB, and GR and OA benefit much more than the others. This makes sense: these functions are annotated on POS tags, and hence perfect tagging will find the correct location of constituents having these labels. The categories MNR and MO essentially mimic attachment decisions. For example, an PP which modifier an NP gets the MNR label, whereas if it modifies a verb it gets the MO label. Given that PP attachment is quite difficult, it is not surprising that scores for these function types are quite low, even with perfect tags.

6.3 Dependencies

Dependencies are quickly becoming a 'new standard' for measuring parser accuracy (cf. Carroll et al., 2002). In addition to the benefits mentioned above, dependencies also help with some evaluation problems we saw in Section 6.2. In particular, using dependencies also allows us to measure the accuracy of grammatical functions independently of node labelling. According to NEGRA annotation, GFs are actually labels for edges (i.e. dependencies) rather than nodes. Therefore, if we include GF labels in the dependency evaluation and leave them out of labelled bracketing evaluation, we can evaluate the accuracy of GF labels without having to cope with the problems of tying GF labels to node labels, as we did in Section 6.2.

To measure dependency accuracy, we must first turn a parse tree into a dependency tree. We do this in a two step process:

i. Annotate each node with its head word. We find the head word using the same approach as in Chapter 3.

ii. For a rule P $\rightarrow$ C_0 ... C_n, let $h(C_i)$ be the head word of the ith child and let h be the head word of the rule. Then, create a dependency for each non-head child. The dependency is a two-tuple of head and dependent. Strictly speaking, for a child j (with $j \neq h$), the dependency is $<h(C_h), h(C_j)>$.

Note that no dependencies are created for unary rules. Therefore, in a sentence with n words, there are always $n-1$ dependencies, regardless of the structure of the tree. During testing, we compute the dependency tree for the gold standard and the parser's guess. A dependency is correct if it appears in both the gold standard and the parser's guess. We measure precision, recall and F-score in the normal way.

At first, it might appear straightforward how to extend the two-tuple definition to include labelled dependencies. If $f(C_i)$ (f is for *function*) is the GF label of the ith child, then a labelled dependency might be defined as $<h(C_h), h(C_j), f(C_j)>$. Unfortunately, such a definition may exclude some GFs from a tree. Recall that in Step ii. above, we disallowed dependencies between a parent and its head child. However, such dependency links do have GF labels. In some of these cases, the GF is simply HD (meaning 'head'), but we do wish to measure if these labels are correct. Therefore, we revise Step ii. to include head children (in more formal terms, the restriction that $j \neq h$ is removed). Overall, this gives us three cases:

Unlabelled dependencies. With the original formulation of Step ii. and treating a dependencies as two-tuples $<h(C_h), h(C_j)>$.

Extended Unlabelled dependencies. With the revised formulation of Step ii. and treating a dependencies as two-tuples $< h(C_h),\ h(C_j) >$. For brevity's sake, we will henceforth refer to these dependencies as 'extended dependencies'

Labelled dependencies. With the revised formulation of Step ii. and treating dependencies as three tuples $< h(C_h), h(C_j), f(C_j) >$.

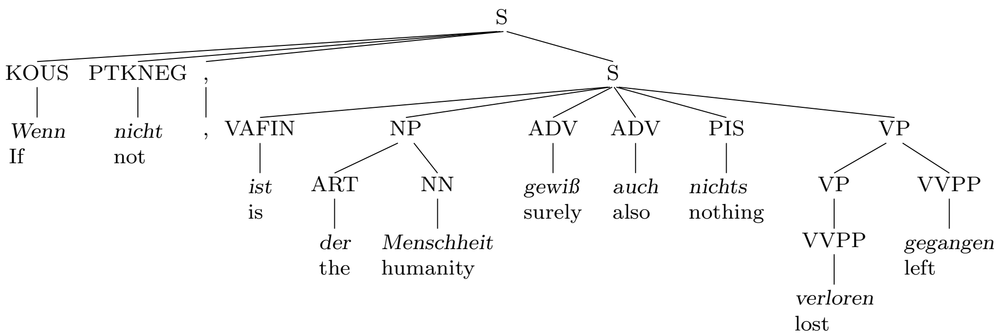
6.3.1 Results

Table 6.8 details the results. The Baseline model scored 80.5 on unlabelled dependencies and 83.8 on extended unlabelled dependencies. Adding GFs to the model increased the unlabelled dependency score to 81.5 and the extended unlabelled dependency score to 84.1. The Sister-head model was able to improve upon these results, managing 81.9 on unlabelled and 85.4 on extended dependencies. The Smooth+GF was again higher with 84.0 on unlabelled and 85.5 on extended dependencies. Quizzically, the Smooth+GF model did not score higher than the Baseline+GF model on labelled extended dependencies, achieving only 77.9 versus 79.7. The highest results overall were due to the model with perfect tags, which achieved 89.1 on unlabelled, 90.6 on extended and 87.6 on labelled dependencies. It is important to remember that the model with perfect tags is not comparable to the other models. As seen in Section 6.1, perfectly disambiguating POS tags is extremely useful in disambiguating parsing ambiguity.

6.3.2 Discussion

In Chapter 4, we saw that the smoothed GF model has a higher labelled bracketing score than the sister-head model. Intriguingly, the sister-head model nearly matches the smoothed GF model on extended unlabelled dependencies. Schiehlen (2004) also noted that dependency accuracies sometimes behaves differently than labelled bracketing scores. Goodman's maximizing metrics Goodman (1998) offer an explanation for the discrepancy between dependency and bracketing scores. Unlexicalized grammars only know about node labels, hence that is all they can emphasize; lexicalized grammars also have knowledge of word-word dependencies, and hence they can put greater emphasis on this these types of features. Gildea (2001) and Bikel (TODO) note that the bilexical grammar of Collins (1997), word-word dependencies do not make a large contribution to overall results. However, in both cases, only results on PARSEVAL were measured. It is unclear if the minor effect of removing word-word dependencies would also be present in dependency measures.

Overall, dependency scores do tend to be higher than bracketing scores, justifying behaviour noted by Lin (1995). To illustrate this, consider the following tree :



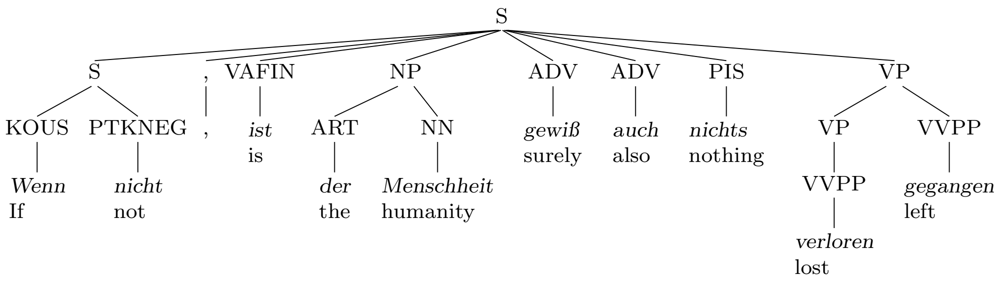
	Unlabelled	Extended	Labelled
Baseline	80.5	83.8	--
Baseline+GF	81.5	84.1	79.7
Sister-head	81.9	85.4	--
Smooth+GF	84.0	85.5	77.9
Smooth+GF+Perfect Tags	89.1	90.6	87.6

Table 6.8. Dependency scores

	POS	LB	Dep	ExDep	LDep
Baseline	97.1	75.0	83.2	86.0	--
Baseline+GF	97.4	76.9	85.4	87.5	82.8
Sister-head	--	77.4	86.6	90.5	--
Smooth+GF	97.4	78.0	85.4	86.7	79.7
Thread	97.4	79.5	86.2	87.7	80.7
Smooth+GF+Perfect Tags	--	86.0	89.2	91.2	88.0

Table 6.9. Performance of various parsers the TIGER corpus

The meaning of the sentence is roughly, “If not, nothing has been lost to humanity?” The Smooth+GF parser does not return the correct parse. Instead, it suggests the following tree:



Basically, the clausal object in the correct parse becomes the main clause in the incorrect parse. This can be considered to be only one error, but $\frac{1}{3}$ of the labelled brackets are wrong. There is only one dependency mistake: *wenn* is modified by *ist* instead of vice versa.

Dependency measures may also be higher than labelled bracketing measures due to other reasons. Recall from Section 6.1 that some parsing errors were due to missed or extra unary nodes. While errors with unary nodes affect labelled bracketing scores, they do not affect standard dependency measures (because unary dependencies are explicitly ignored).

There also are rare instances where the labelled bracket scores were higher than dependency scores. In some instances, the correct head was mislabelled or missing altogether. Picking the wrong head can lead to cascading errors in dependency scores.

6.4 Evaluation on TIGER

The TIGER corpus is designed to supersede NEGRA. It uses a similar annotation format as NEGRA, so it is likely that parsers developed for NEGRA would also perform well on TIGER. Moreover, the TIGER corpus is twice the size of NEGRA. Therefore, one would expect that sparse data would be less of an issue, and therefore performance ought to increase.

We ran six different parsers on the TIGER corpus: the unlexicalized baseline, the baseline with GFs, the sister-head parser, the Smooth+GF parser, the Smooth+GF parser with perfect tags, and the gap-threading model from Chapter 5.

6.4.1 Results

The results of the experiments are listed in Table 6.9. The column headings are as follows. POS refers to POS tagging accuracy, LB refers to labelled bracket *F*-score, Dep refers to unlabelled dependency accuracy, ExDep to extended dependency accuracy, and LDep to the accuracy of dependencies labelled by GFs.

The POS tagging results of all models are competitive with each other, and all are a bit less than 1% better than the results on NEGRA. Labelled bracketing scores are between 1-3% better than on NEGRA. Table 6.9 lists the parsers by increasing labelled bracketing scores on NEGRA. This ranking remains accurate on TIGER, i.e. the best performing parser on NEGRA is also best on TIGER, the 2$^{\text{nd}}$ best on NEGRA is also the 2$^{\text{nd}}$ best on TIGER, etc. However, the boost due to the larger training set was not the same for all models.

The labelled bracketing scores of the baseline improved by 2.5%, from 72.5 to 75.0. The Baseline+GF model saw an improvement of 3.8%, from 73.1 to 76.9. The difference in the sister-head model was almost as high, from 74.1 to 77.4, for a gain of 3.3%. The score of the Smooth+GF model went up by 1.7% to 78.0. Adding threading to the Smooth+GF model improved results to 79.5, a 2.1% gain over the results on NEGRA. Using perfect tags resulted in a labelled bracket score of 86.0%, a 0.8% improvement over the 85.2 seen on NEGRA. As before, we do not report POS tagging results for the sister-head model because of its mix of perfect and guessed tags.

In Section 6.3, we noted that, in NEGRA, a ranking based on dependency accuracy was consistent with the ranking based on labelled bracketing scores. The same is not true of TIGER. After the Smooth+GF+Perfect Tag model, the sister-head parser achieves the highest dependency measure, of 86.6, higher than the 85.4 of the Smooth+GF model. Similar to the finding on the NEGRA corpus, the Smooth+GF model scores lower on labelled dependencies than the equivalent model without smoothing. The Smooth+GF model achieves 79.7 versus 82.8 for the Baseline+GF. Unlike NEGRA, though, the Smooth+GF model also scored lower on unlabelled and extended dependencies.

6.4.2 Discussion

Although there appears to be some variance in the dependency results, it is possible to identify a clear trend in the data. Sparse data is most apparent in models without smoothing (the Baseline and Baseline+GF models) or with lexicalization (the sister-head model), and these models get the biggest boost from the larger training set. Although the Smooth+GF still achieves the best performance on labelled bracketing, the extra data allows the sister-head model to do better on dependencies. This gives more evidence that a simplistic version of Goodman's maximizing metrics are at play here: the sister-head parser has more data to estimate word-word dependencies, and hence gets even more of them right. The Smooth+GF model is still optimized for labelled bracketing, and hence still excels on that metric.

The Smooth+GF+Perfect Tags model had the smallest increase in performance relative to NEGRA, with less than half the gain of the Smooth+GF model on labelled brackets. Because the Smooth+GF model had to guess tags whereas the Perfect Tags model did not, this result suggests that the larger training set helped more with guessing tags than it did with estimating rule probabilities. In other words, the Smooth+GF model did not face as much sparse data in the grammar as it did in the lexicon. Consequently, it may be possible to include more attributes to the Smooth+GF model to solve some of the attachment errors discussed in Section 6.1.2.2.

6.5 Conclusions

There are two primary results from this chapter. First, in Section 6.1, we learned that POS tagging can have a profound impact on parsing results. Many POS tagging errors are simply due to sparse data in the POS tag lexicon. Particular problems are adjective/verb ambiguities and verb/verb ambiguities.

Second, in Sections 6.3 and 6.4, we found that dependency measures behave differently than labelled bracketing measures. Parsers can be made to maximize one type of measure over another. Our results suggest that lexicalized parsers do better on dependencies whereas unlexicalized parsers do better on labelled bracketing.

Overall, it is worth pointing out that the Smooth+GF parser not only claims the highest labelled bracketing scores known to us on the NEGRA and TIGER corpora, but also the highest unlabelled dependency scores of any parser which is not given extra information.

Chapter 7
Conclusions

In statistical parsing research, it is more common for the development new models and discovery of useful linguistic features to occur in English, and only then be applied to other languages rather than vice versa. Indeed, a standard pattern of parsing in with new treebanks is to adapt fully developed English parsing models to the other language. In this dissertation, however, we suggest that linguistic and annotation differences mean that complex models behave in unpredicatble ways. For example, in Chapter 3 we show that English lexicalized models cannot outperform an unlexicalized baseline in the German NEGRA corpus. A closer inspection shows the problem is partially due to annotation differences, and partially due to a lower type/token ratio (i.e. more productive morphology) in German. With this as a starting point, we take a closer look at the effect of annotation and linguistic differences in Chapters 4 and 5. The linguistic differences are the more crucial of the two as the annotation differences between the Penn Treebank and NEGRA were primarily designed to better express linguistic differences between English and German.

A fundamental change is the inclusion of grammatical functions, which we describe in Chapter 4. To gain the full benefit from grammatical functions, we introduce several automatic modifications which improve the parser's ability to model the German case system. Seeing that adding one attribute (in the form of grammatical function tags) is helpful, in Chapter 5, we investigate several others, including a more involved model of morphology and an account of long-distance dependencies. We find that morphological attributes are too noisy to be helpful with our supervised learning approach; other work using unsupervised learning benefits more from morphological information (Beil et al., 1999). While using morphological attributes led to mixed results, there were clear benefits from a pilot parser which could handle long-distance dependencies.

The best performing realistic model uses both smoothing and grammatical functions (henceforth referred to as the Smooth+GF parser). It sets the state-of-the-art performance on the NEGRA and TIGER corpora, with labelled bracketing scores of 76.2 on NEGRA and 79.5 on TIGER. Furthermore, the parser scores 84.0 on dependencies on the NEGRA corpus, also the best reported performance on that corpus, and 86.2 on the TIGER corpus. The sister-head lexicalized parser sets the state-of-the-art dependency score on the TIGER corpus, with a result of 86.6.

7.1 Lessons Learned

7.1.1 Language Matters

One of the fundamental purposes of this thesis was to show that it pays to account for language-specific properties. We benefited from a model of case and aspects of freer word order. But this is not the end of the story. Based on the detailed error analyses of Chapter 6, it is possible to take a closer look at what kind of mistakes parsers are prone to making. We have summarized the result of the error analyses in Table 7.1, re-grouping errors in to more general categories. Some mistakes are unsurprising: the attachment of modifiers such as prepositional phrases is also difficult in other languages. But other errors are due to phenomena not present in languages such as English, including case mistagging and misinterpreting verb conjugations (especially verbs with the -en ending, which are ambiguous between finite and infinite verbs, as wells plural nouns and forms of declined adjectives). Not only have language-specific concerns been important in the models we develop, but because they affect the errors of the models, they ought to play a role in future research.

Model	Error Subtype	Percent of Errors	*F*-Score
Baseline performance			76%
Performance without GF/case errors on POS tags			80%
(But POS tagging errors remain)			
	POS Tagging Errors		
	Verb or adjective ending ambiguity	41%	
	Closed class words	41%	
	Other open class words	16%	
Performance without any POS tagging errors			88%
(But attachment errors remain)			
	Attachment Errors		
	PP attachment	30%	
	Other modifier attachment	9%	
	Co-ordination	15%	
	Annotation mistakes	19%	
	Other attachment errors	27%	

Table 7.1. Where are the errors?

There is a danger of overusing the term ‘language-specific’. It is important to remember that German is not the only language with a case system, or a complex system of verb morphology. Indeed, it is precisely because these are features of other languages that we can claim that our results will be generalizable to other languages with similar properties to German.

7.1.2 Baselines Matter

In Chapter 3, we initially found that the baseline model performed better than a complex lexicalized model. We also noted that parsing research in new languages tend to begin with relatively involved lexicalized models. However, the lesson learned here, complemented by research in Korean (Chung and Rim, 2004) is that it is difficult to predict how well complicated models work in new languages. As a model becomes more complex, it becomes more likely that it ‘overfits’ to one particular language. Therefore, it is better to use a simple model as a baseline. We have suggested unlexicalized PCFGs as a possible ‘universal’ baseline, as it appears to work well in both phase structure and dependency-style treebanks.

7.1.3 Smoothing Matters

Smoothing is a useful tool to improve performance. Improperly evaluated, however, it may lead to ignoring useful features or overemphasizing unimportant ones. In Section 4.3, we found that different smoothing algorithms reacted differently to the same transformations. Broadly speaking, though, the results across smoothing algorithms were correlated: if including a new feature produced a large positive change with one smoothing algorithm, it would do so with another. However, while small changes in results with one smoothing algorithm caused small changes in others, the changes were not always in the same direction.

7.1.4 Evaluation Matters

In Section 6.3, we found that the sister-head and GF parser have a somewhat paradoxical relationship. While the GF parser performs better on labelled bracketing, with sufficient training data, the sister-head parser does better on dependency measures. We hypothesized that Goodman’s maximizing metrics (Goodman, 1998) might provide an explanation: the lexicalized parser includes word-word dependencies in the probability it maximizes, and therefore does better at an evaluation metric based upon word-word dependencies. Unlexicalized parsers, on the other hand, include more information about syntactic categories, and hence does better at labelled bracketing, which places a prime importance on getting the syntactic categories right.

We further hypothesize this is an important result to add to the on-going debate about the importance of lexicalization. Gildea (2001) and Bikel (2004b) found that bi-lexical probabilities add very little to the labelled bracketing scores of Collins' parser. Johnson (1998) and Klein and Manning (2003) show that unlexicalized parsers can do very well on LB measures. On the other hand, later work by Bikel (2004a) shows that bi-lexical probabilities are important in determining the best parse under the Collins models. Moreover, Hockenmaier (2003) reports that bi-lexical probabilities give a substantial boost to the performance of her parser -- using dependency measures as a metric. We leave it to future research to see if (a) removing bilexical probabilities from the Collins model has a bigger effect on dependencies and (b) if the unlexicalized Klein and Manning (2003) grammar outperforms some lexicalized parsers on dependencies, as it does on labelled brackets.

7.2 Future Work

Based on the success of including case in the unlexicalized parsing models in Chapter 4, we developed a grammar which included a more complete treatment of the morphology of nouns and noun dependants in Chapter 5. While this did not prove to be successful, the error analysis in Chapter 6 suggests that a better analysis of the morphology of *verbs* might be more useful than an improved model for nouns. In particular, the Smooth+GF parser had some difficulty with verb tense ambiguity as well as verb/adjective ambiguity.

The morphological analysis was somewhat incomplete in that it did not attempt to make use of stemmed word forms. This might especially have an impact on lexicalized parsing, as we found in Chapter 6 that sparse data does have a profound impact on the sister-head parsing model. Moreover, as we found that the unlexicalized and lexicalized parsing models had different strengths, one relatively large area for further research is to recombine the two models. The approach used by Charniak (1997), and its extension in Charniak (2000) might be useful for such a recombination as it allows the unlexicalized and lexicalized components to be split. Although we found the parameterization of Charniak (1997) to be unsatisfactory for in the NEGRA corpus (Chapter 3), a sister-head version of this model might prove successful.

We included a treatment of crossing dependencies in part to aid with the recovery of sentences exhibiting scrambled constituents. To test the success of this, we investigated the effect of crossing dependencies on some free word order constructions like topicalization. However, a more complete evaluation of crossing dependencies would be both possible and interesting.

7.3 Final Words

In this thesis, we have developed an accurate broad-coverage parser for German. We found that case and word order did have a strong influence on parsing results. Indeed, our models benefitted from including a simple yet effective model of case as well as a preliminary model of crossing dependencies. These results suggest a set of features useful for building parsers for languages similar to German, and they may also inform the design of automatic methods for treebank engineering (Chiang and Bikel, 2002). This work also lead to several practical insights. For example, we found the choice of evaluation measures and smoothing may obscure or overrate the importance of linguistic cues. We also found, contrary to custom, that using a baseline is necessary when developing parsers in new and untested languages.

Category	Search Direction	
AA	Right	ADJA ADJD PIS
AP	Right	ADJA ADJD CAP CARD PIAT PIDAT PIS
AVP	Right	ADV AVP CAVP ADJA ADJD
CH	Left	Leftmost daughter
DL	Right	S CS
ISU	Right	Leftmost daughter
MPN	Right	NE FM CARD
MTA	Right	ADJA NE TRUNC
NM	Right	NN PIAT CARD ORD N?
NP	Right	NN NE NM MPN NP CNP PRELS PWAT PWS PDS PDAT PIS PIAT PIDAT PPER PPOSS PPOSAT
PP	Left	APPR APPRART APPO APZR CA
QL	Right	Rightmost daughter
S	Right	VVFIN VMFIN VAFIN VVIMP VAIMP VMPP VVPP VP CVP S CS VVINF VAIMP NP PP
VP	Right	VVPP VVINF VAINF VMINF VVIZU VAPP VZ CVZ VP CVP AVP AP
VZ	Right	VVINF VAINF VMINF ADJA PPOSAT

Table A.1. Head finding rules for standard categories

Appendix A
Head-finding Rules

The concept of lexicalization is intrinsically linked to the concept of headedness. The lexicalized parsing models from Chapter 3 propagate lexical elements up the tree based upon which element is the syntactic head of the phrase. The -HD (head) grammatical function denotes that a daughter is the head of a constituent. However, this grammatical function is only used with a few constituents, such as the S, VP and AP categories. An alternative approach is necessary for other constituents.

A common technique for finding heads, proposed by Magerman and detailed by [Collins(1999)] is to use head-finding rules. Given a parent and a list of children, these rules select which daughter among the children is the head. There is a distinct rule for each parent category. The rules specify a list of syntactic categories which may serve as the head child for the given parent. The rules also define whether to search for the head daughter from left-to-right (denoted 'Left' in the table) or from right-to-left (denoted by 'Right'). The general strategy for finding the head is to search for the first child in the list in the specified direction. If it is no match is found, then the first child is picked as the head. This would be the leftmost child if searching from left-to-right and the rightmost if searching in the opposite direction.

Table A.1 summarizes the head-finding rules for the non-co-ordinated categories. A list of the head-finding rules for co-ordinated categories is found in Table A.1. Wherever possible, the rules are quite similar to the rules developed by Magerman for English. While Magerman's rules are quite widely used, they have never been truly evaluated. However, because some categories in NEGRA do carry annotations specifying the head child, it is possible to test how often the head-finding rules agree with the annotated heads. We found that they agreed upward of 98% of the time.

CAC	Left	APPR APZR APPO
CAP	Left	ADJA ADJD CARD
CAVP	Left	AVP ADV CAVP
CCP	Left	Leftmost daughetr
CNP	Left	NP NN NE MPN CNP CARD PRF PIS PPER FM
CPP	Left	PP CPP
CVP	Left	VP VZ CVP VVINF VVPP VVIZU
CVZ	Left	VZ CV
CS	Left	S CS CO D?
CO	Left	S VVINF VVPP VP NP PP AP ADJA ADJD PI?

Table A.2. Head finding rules for co-ordinated categories

Bibliography

Steven Abney. Stochastic Attribute-Value Grammars. *Computational Linguistics*, 230 (4):0 597--618, 1997.

Abhishek Arun. Statistical Parsing of the French Treebank. Master's thesis, University of Edinburgh, 2004.

Markus Becker and Anette Frank. A Stochastic Topological Parser of German. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 71--77, Taipei, 2002.

Franz Beil, Glenn Carroll, Detlef Prescher, Stefan Riezler, and Mats Rooth. Inside-Outside Estimation of a Lexicalized PCFG for German. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, University of Maryland, College Park, 1999.

Franz Beil, Detlef Prescher, Helmut Schmid, and Sabine Schulte im Walde. Evaluation of the Gramotron Parser for German. In *Proceedings of the LREC Workshop Beyond Parseval: Towards Improved Evaluation Measures for Parsing Systems*, Las Palmas, Gran Canaria, 2002.

Daniel M. Bikel. A Distributional Analysis of a Lexicalized Statistical Parsing Model. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 184--189, 2004a.

Daniel M. Bikel. Intricacies of Collins' Parsing Model. *Computational Linguistics*, 2004b. To appear.

Daniel M. Bikel and David Chiang. Two Statistical Parsing Models Applied to the Chinese Treebank. In *Proceedings of the 2nd ACL Workshop on Chinese Language Processing*, Hong Kong, 2000.

Ezra Black, Frederick Jelinek, John D. Lafferty, David M. Magerman, Robert L. Mercer, and Salim Roukos. Towards history-based grammars: Using richer models for probabilistic parsing. In *Meeting of the Association for Computational Linguistics*, pages 31--37, 1993.

Ezra Black, John D. Lafferty, and Salim Roukos. Development and Evaluation of a Broad-Coverage Probabilistic Grammar of English-Language Computer Manuals. In *Proceedings of the 30th Meeting of the Assosication for Computational Linguistics*, pages 185--192, Newark, DE, 1992.

Don Blaheta and Eugene Charniak. Assigning function tags to parsed text. In *Proceedings of the 1st Conference of the North American Chapter of the ACL (NAACL), Seattle, Washington.*, pages 234--240, 2000.

Rens Bod. An Efficient Implementation of a New DOP Model. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 19--26, Budapest, 2003.

T.L. Booth and R.A. Thompson. Applying Probability Measures to Abstract Languages. *IEEE Transactions on Computers*, C-22(5):0 442--450, 1974.

Thorsten Brants. Cascaded Markov Models. In *Proceedings of the 9th Coference of the European Chapter of the Association for Computational Linguistics EACL-99*, pages 117--125, 1999.

Thorsten Brants. TnT: A statistical part-of-speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle, 2000.

Thorsten Brants and Matthew Crocker. Probabilistic parsing and psychological plausibility. In *Proceedings of the 18th International Conference on Computational Linguistics COLING-2000*, Saarbrücken/Luxembourg/Nancy, 2000.

Eric Brill. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, 210 (4):0 543--565, 1995.

Peter Brown, Vincent Della Pietra, Peter deSouza, Jennifer Lai, and Robert Mercer. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, Volume 180 (Number 4):0 466--479, December 1992.

Glenn Carroll and Mats Rooth. Valence induction with a head-lexicalized PCFG. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 36--45, Granada, 1998.

John Carroll, Anette Frank, Dekang Lin, Detlef Prescher, and Hans Uszkoreit, editors. Las Palmas, Gran Canaria, 2002.

Eugene Charniak. Tree-bank grammars. Technical Report CS-96-02, Department of Computer Science, Brown University, 1996.

Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proceedings on the Fourteenth National Conference on Artificial Intelligence*, Menlo Park, CA., 1997.

Eugene Charniak and Sharon Caraballo. New Figures of Merit for Best-First Probabilistic Chart Parsing. *Computational Linguistics*, 240 (2):0 275--298, 1998.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, 1998.

David Chiang and Daniel M. Bikel. Recovering Latent Information in Treebanks. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 183--189, Taipei, 2002.

K.S. Choi. Kaist language resources. Technical report, Korean Advanced Institute of Science and Technology, 2001.

Noam Chomsky. *Lectures on Government and Binding: The Pisa Lectures*. Walter de Gruyter Inc, Berlin and New York, 1981. Reprint. 7th Edition.

Hoojung Chung. *Statistical Korean Dependency Parsing Model based on the Surface Contexual Information*. PhD thesis, Korea University, January 2004.

Hoojung Chung and Hae-Chang Rim. Unlexicalized Dependency Parser for Variable Word Order Languages based on Local Contextual Pattern. In *Computational Linguistics and Intelligent Text Processing (CICLing-2004)*, pages 112--123, Seoul, Korea, 2004.

Michael Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184--191, Santa Cruz, CA, 1996.

Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

Michael Collins and Nigel Duffy. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of the 40th Conference of the Association for Computational Linguistics*, 2002.

Matthew Crocker and Thorsten Brants. Wide coverage probabilistic sentence processing. *Journal of Psycholinguistic Research*, 290 (6):0 647--669, 2000.

Walter Daelemans, Antal Van Den Bosch, and Jakub Zavrel. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34, 1999.

N. M. Dempster, A.P. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society of Britain*, 39:0 185--197, 1977.

Péter Dienes and Amit Dubey. Antecedent recovery: Experiments with a trace tagger. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 33--40, Sapporo, Japan, 2003a.

Péter Dienes and Amit Dubey. Deep processing by combining shallow approaches. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 431--438, Sapporo, Japan, 2003b.

Amit Dubey and Frank Keller. Parsing German with Sister-head Dependencies. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 96--103, Sapporo, Japan, 2003.

Abdessamd Echihabi and Daniel Marcu. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, 2003.

Jason Eisner. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 340--345, Copenhagen, 1996.

Sisay Fissaha, Daniel Olejnik, Ralf Kornberger, Karin Müller, and Detlef Prescher. Experiments in German Treebank Parsing. In *Proceedings of the 6th International Conference on Text, Speech and Dialogue (TSD-03)*, Ceske Budejovice, Czech Republic, 2003.

Anette Frank, Markus Becker, Berthold Crysmann, Bernd Kiefer, and Ulrich Schäfer. Integrated Shallow and Deep Parsing: TopP meets HPSG. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 104--111, Sapporo, Japan, 2003.

Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. *Generalized Phase Structure Grammar*. Basil Blackwell, Oxford, England, 1985.

Dan Gildea. Corpus Variation and Parser Performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 167--202, Pittsburgh, PA, 2001.

Joshua Goodman. *Parsing inside-out*. PhD thesis, Harvard University, 1998.

Erhard Hinrichs and Tsuneko Nakazawa. Linearizing AUXs in German verbal complexes. *German in Head-driven Phrase Structure Grammar*, pages 11--37, 1994. Lecture Notes No. 46.

Julia Hockenmaier. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. PhD thesis, Institute for Communicating and Collaborative Systems, School of Informatics, University of Edinburgh, 2003.

Tilman Höhle. Der Begriff Mittelfeld, Anmerkungen über die Theorie der topologischen Felder. In *Akten des 7. Internationalen Germanisten-Kongresses, Göttingen 1985*, volume 4 of *Kontroversen, alte und neue*, pages 329--340, 1986.

Frederick Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands, May 1980.

Mark Johnson. *Attribute-Value Logic and the Theory of Grammar*. CSLI Publications, 1988.

Mark Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 240 (4):0 613--632, 1998.

Mark Johnson, Stuart Geman, Stephan Canon, Zhiyi Chi, and Stefan Riezler. Estimators for Stochastic ‘‘Unification-Based” Grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, University of Maryland, College Park, 1999.

Mark A. Jones and Jason M. Eisner. A Probabilistic Parser Applied to Software Testing Documents. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-92)*, pages 322--328, San Jose, CA, 1992.

Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(3):0 400--401, March 1987.

Dan Klein and Christopher D. Manning. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423--430, Sapporo, Japan, 2003.

Reinhard Kneser and Hermann Ney. Improved back-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 181--184, Amsterdam, The Netherlands, May 1980.

Sandra Kübler. Parsing without grammar -- using complete trees instead. In *Proceedings or RANLP 2003*, Borovets, Bulgaria, 2003.

K. J. Lee. *Probabilistic Parsing of Korean based on Language-Specific Properties*. PhD thesis, Korea Advanced Institute of Science and Technology, 1997.

Roger Levy and Christopher D. Manning. Deep Dependencies from Context-Free Statistical Parsers: Correcting the Surface Dependency Approximation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 2004.

David M. Magerman. Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276--283, Cambridge, MA, 1995.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 190 (2):0 313--330, 1993.

John Nerbonne. Partial verb phrases and spurious ambiguities. *German in Head-driven Phrase Structure Grammar*, pages 109--150, 1994. Lecture Notes No. 46.

Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:0 1--38, 1994.

Carl J. Pollard and Ivan Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994.

Detlef Prescher. Inside-outside estimation meets dynamic EM. In *Proceedings of the 7th International Workshop on Parsing Technologies (IWPT-01)*, Beijing, China, 2001.

Sabine Schulte im Walde. The German Statistical Grammar Model: Development, Training and Linguistic Exploitation. Linguistic Theory and the Foundations of Computational Linguistics 162, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, December 2000.

Anoop Sarkar and Chung hye Han. Statistical morphological tagging and parsing of Korean with an LTAG grammar. In *Proceedings of the Sixth Workshop on Tree Adjoining Grammars*, Venice, May 2002.

Micheal Schiehlen. Combining Deep and Shallow Approaches in Parsing German. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 2003.

Micheal Schiehlen. Annotation Strategies for Probabilistic Parsing in German. In *Proceedings of the 20th International Conference on Computational Linguistics*, 2004.

Helmut Schmid. Improvement in Part-of-Speech Tagging with an Application to German. In *Proceedings of the ACL SIGDAT-Workshop*, March 1995.

Helmut Schmid. LoPar: Design and implementation. Technical Report 149, Institute for Computational Linguistics, University of Stuttgart, 2000.

Helmut Schmid. A Generative Probability Model for Unification-Based Grammars. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002)*, Taipei, 2002.

Helmut Schmid and Sabine Schulte im Walde. Robust German Noun Chunking with a Probabilistic Context-Free Grammar. In *Proceedings of the 18th International Conference on Computational Linguistics*, Saarbrücken, Germany, August 2000.

Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar*. CSLI Publications, 1986.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. An annotation scheme for free word order languages. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, Washington, DC, 1997.

Andreas Stolcke. *Bayesian Learning of Probabilistic Language Models*. PhD thesis, University of California at Berkeley, 1994.

Tylman Ule. Directed Treebank Refinement for PCFG Parsing. In *Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories (TLT 2003)*, Växjö, Sweden, November 2003.

Hans Uszkoreit. *Word Order and Constituent Structure in German*. CSLI Publications, Stanford, CA, 1987.

Ian H. Witten and Timothy C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):0 1085--1094, July 1991.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics and the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 523--530, Toulouse, 2001.

?