Statistical Parsing with Context-Free Filtering Grammar

by

Michael Demko

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Statistical Parsing with Context-Free Filtering Grammar

Michael Demko

Master of Science

Graduate Department of Computer Science

University of Toronto

2007

Statistical parsers that simultaneously generate both phrase-structure and lexical dependency trees have been limited in two important ways: the detection of non-projective dependencies has not been integrated with other parsing decisions, or the constraints between phrase-structure and dependency structure have been overly strict. I develop context-free filtering grammar as a generalization of the more restrictive lexicalized factored parsing model, and I develop for the new grammar formalism a scoring model to resolve parsing ambiguities. I demonstrate the flexibility of the new model by implementing a statistical parser for German, a freer-word-order language exhibiting a mixture of context-free and non-projective behaviours.

Dedication

For Miriam, John and Alana

Acknowledgements

I would like to thank my supervisor, Gerald Penn, for his support and assistance, and especially for introducing me to the very interesting problems of freer-word-order parsing. With respect to the thesis itself, Gerald has been particularly helpful in the area of formalizing my ideas (in Chapter 3) and I greatly appreciate his comments and recommendations about the document in general.

The thesis has also benefited greatly from the comments of Suzanne Stevenson who, in addition to being an unusually careful reader with a commitment to scientific rigour, has a talent for suggesting those changes that most substantially improve the readability of a text.

I am grateful to both Suzanne and to Graeme Hirst for their guidance on some occasions while Gerald was on sabbatical. The students of the computational linguistics group have also been very supportive, and I would like to single-out Saif Mohammad for his constant encouragement and timely advice. Outside of school, Elham Marzi and Premek Bakowski saw me through some dark times: without their friendship this thesis may not have been completed.

Finally, I would like to acknowledge the Ontario Graduate Scholarship Program for its financial support of this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

If we are ever to realize the dream of natural language interaction with computational systems, or of automatic manipulation of the highly structured information encoded in natural language resources, we must first find a way to automatically determine the semantic content of sentences. As a first approximation to determining the structural aspects of semantics (which entities relate to which others, and how), many researchers have looked at the problem of discovering the syntactic structure of sentences.

Stepping back from the lofty goal of semantic analysis, we can also imagine applications of parsers that already seem within our grasp: parsing is directly applicable to the automatic detection of grammatical errors, and may be useful in machine translation systems. Automatic parsers can also be useful tools for computational linguists desiring approximately correct structural annotations of raw text.

One could also argue that automatic parsing projects are a testing-bed for linguistic theory. The process of encoding any formal model in the rigorously logical languages currently required by computers is bound to uncover many of the ambiguities and contradictions inherent in a product of human imagination. Successes in parser development can also be seen as supporting the claims that their underlying linguistic models provide a basis for feasibly analyzing syntax (a claim that is frequently not amenable to direct mathematical proof.)

Early parsing research aimed (ultimately) to provide a complete model of syntax: one capable of making all possible syntactic distinctions, capable of determining when sentences were in or not in a given language, and capable of determining the syntactic structure of all sentences in the language. The immensity of this task has led to a branching of research, each branch focusing on different aspects of the problem (often at the expense of others.) The particular branch with which I identify this work is *statistical parsing*.

Statistical parsers focus on disambiguation and coverage: determining *one* syntactic structure for each of as many sentences in a language as possible. They usually abandon all other elements of syntactic analysis: in particular they assume that every sentence is grammatical and make no attempt to flag ungrammatical input. The output of a statistical parser is often 'mostly correct', but frequently includes some errors. Statistical parsers generally rely on very simplistic linguistic models, and derive most of their capability from machine learning algorithms trained over large human-annotated corpora.

Within the statistical parsing community, two kinds (broadly speaking) of syntactic analysis are done: phrase structure analysis, often based on context-free grammar; and dependency analysis. *Phrase structure analysis* aims to group words together in progressively larger units (constituents) until one unit is found that contains the entire sentence. In principle, each of the constituents formed in this process has a complete corresponding semantic interpretation, though parsers rarely go so far as to analyze the semantics. *Dependency analysis*, on the other hand, builds a directed tree of binary relations over the words of a sentence. Each relation encodes precisely the role of a dependent word with respect to its governing word. Dependency analyses can be either projective (disallowing crossing dependencies) or non-projective (permitting crossing dependencies.)

The semantic meaning of a dependency tree is perhaps less intuitive than that of a phrase structure tree, but phrase structure analysis has a critical weakness: it is computationally challenging in languages with freer word order and discontinuous constituents. Even the relatively fixed word order of English poses problems for statistical systems aiming to produce trees for semantic analysis (addressing constituent movement has spawned a number of papers on its own.) Non-projective dependency analysis does not have this weakness. It is therefore desirable for a system to be able to do both a simplified phrase structure analysis and a full dependency analysis at the same time.

The primary contribution of this work is a grammar formalism and a statistical parsing system that permit simultaneous context-free phrase structure parsing and non-projective dependency parsing. The system is derived from the factored parsing model introduced by Klein and Manning (2002). Their model is essentially a lexicalized context-free parsing model, except that the lexical statistical model is independent of the rest of the phrase structure model. By relaxing their requirements of correspondence between lexical relations and phrase structure, I am able to propose a model permitting a much broader range of dependency analyses. Ultimately, I combine a phrase structure parser like that of Klein and Manning (2003) with a non-projective dependency parser based on that of McDonald et al. (2005b).

I evaluate the combined parser on the TüBa-D/Z treebank of German newspaper text. The experiment demonstrates that forcing a certain level of agreement between the phrase structure analysis of a sentence and its dependency analysis does not degrade the accuracy of either. The absolute values of the accuracies achieved by all parsers in the study confirm the claim of Kübler et al. (2006) that German newspaper text is just as easily parsed as English newspaper text, provided that the target annotation is well chosen.

1.2 Thesis Statement

I take the position that phrase structure analysis and dependency analysis of the syntax of a sentence are complementary. Both analyses can be performed simultaneously, with reasonable efficiency and accuracy, in a statistical parsing framework.

The specific formalism I propose for relating the two analyses is *context-free filtering grammar*, a CFG-like grammar formalism that specifies which words of a constituent are available to enter into dependency relations with the rest of a sentence.

The specific statistical model I propose is a factored model in the style of Klein and Manning (2002) which combines an unlexicalized context-free model with a non-projective maximum-spanning-tree dependency model (McDonald et al., 2005b).

1.3 Outline of Study

In support of this thesis are the following chapters:

Chapter 2: Related Work This chapter surveys much of the main-stream statistical parsing work of the late 1990s, along with more recent work of relevance to the thesis. The survey pays particular attention to statistical phrase structure parsing in German, to non-projective unlabeled dependency parsing, and to the machine learning techniques used in parsing systems.

Chapter 3: Context-Free Filtering Grammar This chapter defines context-free filtering grammar, examines its relationship to the factored lexicalized parsing framework of Klein and Manning (2002), and provides a parsing algorithm. The chapter also describes the statistical parametrization and A* scoring heuristic used in the implementation of my parser.

Chapter 4: Experimental Design This chapter describes the set-up of an experiment to test the hypothesis that the new model can outperform existing models.

Chapter 5: Experimental Results This chapter provides the results of the experiment. The experiment clearly demonstrates that phrase structure and dependency analysis can be combined without sacrificing accuracy, but does not establish that the new model can outperform older ones.

Chapter 6: Conclusions The final chapter reviews the contributions of this work, reiterates the limitations of this research, proposes some applications of context-free filtering grammar, and highlights some opportunities for refinements of the framework.

1.4 Summary of Contributions

This thesis project contributes the following to the field of computational linguistics:

context-free filtering grammar: The new grammar formalism provides a framework in which to analyze the interaction between phrase structure and dependency structure. It may be of use on its own or, more likely, as an inspiration for more nuanced formalisms.

a parsing algorithm for context-free filtering grammar: The existence of a chart-parsing algorithm for the new type of grammar makes the grammar more accessible to the research community than it might otherwise be, and permits an intuitive understanding of the properties of the new grammar.

an implementation of a parser using context-free filtering grammar: I implemented the parsing system described in this thesis, demonstrating its feasibility.

a TüBa-D/Z -derived dependency treebank: The heuristically extracted dependency treebank used for my experiment could be an excellent starting point for a manually annotated/verified dependency treebank of German.

an evaluation of a context-free filtering grammar model: The results of Chapter 5 demonstrate that the new parsing model is capable of generating phrase structure and dependency analyses that are (independently of one another) just as good as those produced by pure phrase structure or pure dependency parsers.

1.5 Terminology

In order to carefully explain the grammar formalism introduced in this thesis, I will have to make reference to a large number of concepts used by the statistical parsing community. The terminology for some of these concepts is not always consistent between publications, so I would like to define here the terms I use.

Context-free grammar (CFG) and *context-free grammar parsing* are consistently defined concepts in the parsing literature. The trees of nested labeled constituents produced by context-free parsers, I call *context-free phrase structure trees*, or simply *phrase structure trees.*

The statistical variant of context-free parsing to which I will make reference in this thesis is *probabilistic context-free grammar parsing* (PCFG parsing). A *probabilistic context-free grammar* is a CFG that assigns to each production rule a probability, conditional on the label of the rule's left-hand-side symbol. A phrase structure tree derivable by the grammar is defined to have a probability equal to the product of all of the production rules in the tree's derivation.

A refinement on PCFG parsing is *lexicalized PCFG parsing.* The trees output by a lexicalized PCFG parser are called *lexicalized phrase structure trees.* A lexicalized PCFG is a PCFG in which each non-terminal is augmented with a *lexical head* token. I will also use the term *lexical head* to denote the token marked as the head of a constituent in a lexicalized phrase structure tree. The lexical head of the left-hand-side of a preterminal rule must match the terminal on the right-hand-side. The lexical head of the left-hand-side of each other production rule must match the lexical head of one of the right-hand-side symbols of the rule.

Klein and Manning (2002) define a parsing model that also produces lexicalized phrase structure trees, but that does not use a lexicalized PCFG. Instead it uses a PCFG and a separate lexical dependency model. This kind of parser I will call the *factored parser of Klein and Manning (2002).* This factored parsing model is the starting point for my work.

A *lexical dependency tree*, or simply *dependency tree*, is a directed tree over the tokens of a sentence. Each edge in a dependency tree represents a relation between two tokens.[1] The tail of the edge is the *governor.*[2] The destination of the edge is the *dependent. Labeled dependency trees* have edges labeled with syntactic functions, *unlabeled dependency trees* omit this information. I use unlabeled trees in this thesis.

A token x in a dependency tree *dominates* another token y if x is the governor of y, or if x is the governor of x' and x' dominates y.

In a tree over an ordered sequence of tokens, a dependency edge from token x to y is *projective*, if for each token z between x and y, x dominates z; otherwise the edge is *non-projective.* If all of the edges of a dependency tree are projective, then the tree is called projective, otherwise it is called non-projective. The definitions of this paragraph are adapted from Kahane et al. (1998).

In discussing parsing strategies, I will differentiate between a number of closely related ideas. A *grammar* is a formal set of rules defining the allowable syntactic structures of a language (and by association, the allowable utterances of the language.) In some cases a grammar may be nearly trivial, permitting for instance 'all directed trees over any set

[1]The term *edge* should not be confused with the term *arc.* I will use *edge* to refer to dependency relations, and *arc* to refer to partial-parse data-structures in chart parsers.

[2]Some writings refer to the *governor* as the 'parent' or 'head', but this leads to confusion with the term *lexical head* which I reserve to denote the head of a constituent in a lexicalized phrase structure tree. I will also *not* speak of governing with respect to phrase structure; I will only use the term when writing about dependency structure.

of words in the language'.

A *statistical model*, or *scoring model*, is a function mapping a vector of *model parameters* and a syntactic structure to a real number (a *score*).

A *parsing system* is an implementation of a search algorithm that finds the best-scoring syntactic structure that matches an utterance (provided as input) according to a grammar, a scoring model and a vector of model parameters values. Although I will use the term *parser* fairly loosely, in principle it applies to the combination of a specific parsing-system with a specific grammar, a specific scoring model and a specific vector of model parameter values.

Parameter-estimation or *training* is the process of selecting a vector of model parameters that can be expected to maximize the accuracy of a parser.

Chapter 2

Related Work

2.1 Overview

Statistical systems that learn to parse sentences of a particular language (and genre) by training on large manually-parsed corpora have been a topic of interest to the natural language processing community for over a decade. The bulk of the earlier statistical parsing work focused on retrieving *phrase structure* analyses of the syntax of sentences. In this chapter we will review:

- the models used in these phrase structure parsers;
- some of the problems with statistical phrase structure analysis, particularly with respect to German, a freer-word-order language;
- some models used more recently in dependency parsers; and
- general statistical techniques that have been adapted to parsing.

Finally, I will summarize and highlight those themes that are especially relevant to this thesis.

2.2 Probabilistic Context-Free Parsing

Rather than reviewing in detail several projects, I have elected to describe one system in particular, and to point out its relation to a number of others. I have done this primarily to avoid repeating the many similarities between these systems, and to provide as coherent a narrative as I can.

2.2.1 Themes in Probabilistic Phrase Structure Parsing

The Collins parser (Collins, 1999, 2003) is among the best-known statistical parsing projects. We will review the contributions of Collins (2003) (a re-examination of some of his previous work) to phrase structure parsing: his results remain close to state-of-the-art and design themes similar to his arise in other projects.

The Collins parser takes as input part of speech labeled text sentences. The parts of speech are ignored except for words unknown by the parser's language model. The parser outputs phrase structure parse trees that reflect the annotation patterns of the treebank on which the model was trained. Development of the initial models was done on the Penn treebank (Marcus et al., 1993).

The Collins models (Collins proposes 3 of them) can be related to two main traditions: probabilistic context-free grammar (PCFG) parsing and history-based parsing. PCFG parsers model language as a *stochastic branching process* that produces different sentences with different probabilities. The series of decisions leading to the final result encode the corresponding parse tree. Parsing in this paradigm is a matter of discovering the most probable canonical derivation for a given input sentence (maximizing $P(derivation, sentence)$, where *sentence* is fixed.) PCFG models can be created automatically by stripping a context free grammar from a treebank corpus, and estimating rule probabilities by straight-forward arithmetic; the performance of this naïve approach is shown by Charniak (1996) to be surprisingly good. To achieve better results the model can be enhanced by considering dependencies between lexical heads assigned to all constituents (for example Charniak, 1997).

History-based parsing sees a parse as a *sequence of decisions* made in the course of mapping a sentence to a parse tree (Black et al., 1993; Jelinek et al., 1994; Magerman, 1995; Ratnaparkhi, 1997). These decisions can either be sentence generation decisions (as in PCFG) or sentence-specific parsing decisions (i.e. maximizing $P(tree|sentence)$.) Decisions are not considered independently: any decision made earlier in the course of a process can be used as part of a conditioning context for decisions made later. The advantage of history-based models over PCFG models is their ability to make use of a broader range of features.

The Collins models, like PCFG-derived models, concern themselves with top-down language generation decisions rather than with left-to-right parsing decisions, and are history-based in that they do not model choices independently but rather as a sequence.[1] In particular, Collins sees the generation of a sentence as a top-down head-outward stochastic process that, given a phrase category P and its head word h:

1. generates a phrase category H for the head child, with probability $P(H|P, h)$. H is then recursively expanded.

2. generates each child category L_i and head word l_i to the left of the head child, from the head outward, with probability $P(L_i(l_i)|P, h, H, cl_i)$, where cl_i provides access to historical information about previous decisions. Each child is recursively expanded before the next child is generated.

3. generates each child category R_i and head word r_i to the right of the head child, from the head outward, with probability $P(R_i(r_i)|P, h, H, cr_i)$, where cr_i provides access to historical information about previous decisions. Each child is recursively

[1]I address here only the abstract fundamentals of the Collins models. I do not mention the separate language model for 'base' noun-phrases, the handling of punctuation or the nuanced smoothing techniques, all of which contribute significantly to the high accuracies that Collins achieves.

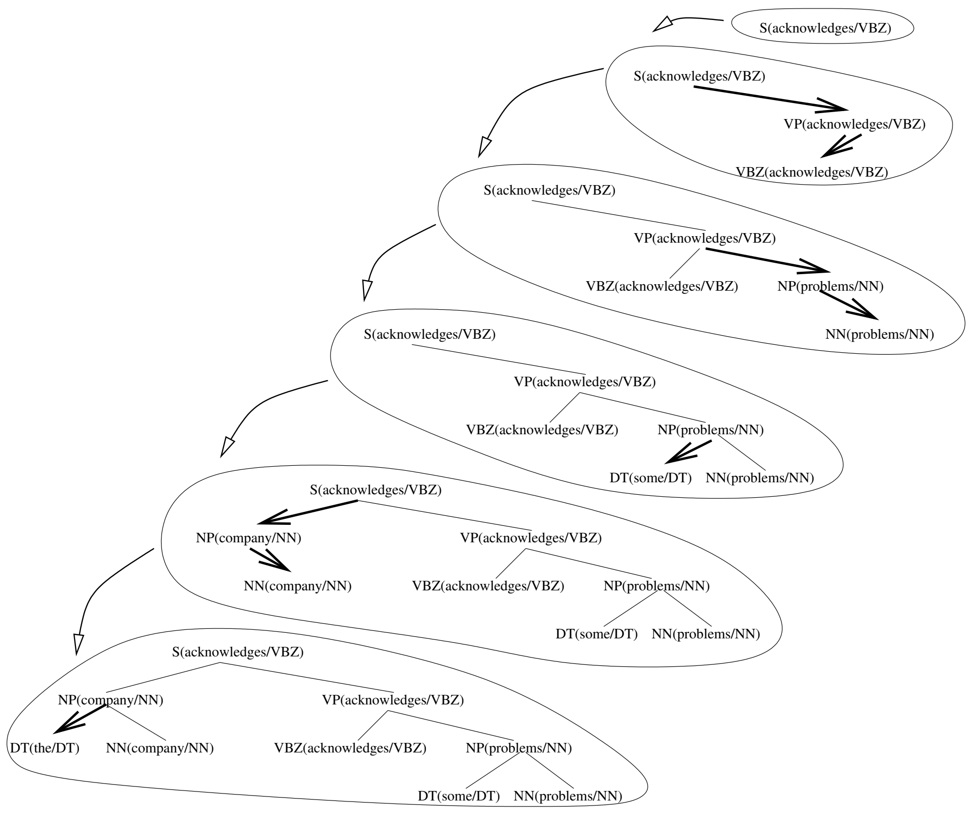

Figure 2.1: Top-down head-outward generation of the sentence “the company acknowledges some problems”, from the Penn treebank II (Marcus et al., 1994)

.

expanded before the next child is generated.

This head-driven language generation model is illustrated in figure 2.1.

In model 1, cl_i and cr_i consist of ‘distance’ information about whether the current child is immediately adjacent to the head and whether it is separated from the head by a verb. In model 2, cl_i and cr_i can include subcategory information for predicate phrase structures. Model 3 will be mentioned below in section 2.3. Parsing sentences of the Penn treebank (of 40 words or less), Model 2 achieves labeled bracketing precision and recall both over 88% with fewer than 0.95 crossing brackets per sentence (these are the commonly used Parseval metrics, Black et al., 1991).

2.2.2 Analyses of Probabilistic Parsing Models

In addition to research that builds up complex generative models like those of Collins, other work has been done to evaluate the weaknesses and critical components of these systems. Gildea (2001) argues that with most of the parsing work being done on Wall-street Journal text, researchers are getting a skewed sense of how useful their statistics can be: the Collins model 1 isn't nearly as good on the broader-domain Brown corpus, for instance. Gildea also raises a surprising question about whether or not modeling lexical dependencies is really achieving all that much (removing bilexical dependencies from model 1 impacts precision and recall by less than 0.5%.) Bikel (2004), in his careful analysis of the Collins models, explains this by pointing out that sparse data almost always forces the models to back off to probability estimates that don't encode bilexical dependencies. He notes, however, that lexical heads *are* useful for predicting the child categories of a constituent.

Work by Johnson (1998) and Klein and Manning (2003) explores the idea of *grammar transformation* to improve the accuracy of unlexicalized PCFG-like models. This technique involves automatically applying transformations to a treebank before extracting a grammar from it, parsing according to the transformed grammar, and then reversing the transformations to acquire trees in the original format. Both works find *parent-encoding* (relabeling each constituent to include the original label of its parent, or even grandparent) to be highly beneficial. In addition to parent-encoding and a number of more fine-grained relabeling schemes, Klein and Manning (2003) also investigate the impact of so-called *Markovization*, the technique of generating child constituents from a head outward (as in the Collins models described above) rather than relying on fixed treebank grammar rules. They found Markovization to be beneficial, but to a lesser extent than parent-encoding. All in all, Klein and Manning are able to achieve an 86.3% F_1 score without resorting to lexicalization, to references to arbitrary elements of generation history (unlike the Collins models), or even to probability estimate smoothing for syntax rules.

2.2.3 Factored Parsing, an Important Digression

The Stanford Parser is strongly based on two papers, the second of which (Klein and Manning, 2003) we have just looked at. In the first paper, Klein and Manning (2002) propose splitting (or factoring) the probability models for phrase structure and lexical dependencies (figure 2.2; figure 2.3 represents the same two trees, emphasizing the constraints between them.[2]) An unlexicalized PCFG model can assign a score to all possible phrase structure trees, a projective lexical dependency model can assign a score to all possible dependency trees, and the optimally scoring combined tree (a phrase structure tree with lexical heads) can be recovered exactly by means of an A* search. The present work builds heavily on this idea of a factored model, but loosens the structural correspondence between constituents and lexical heads in the combined tree.

The factored model of Klein and Manning (2002) is in some sense similar to topological parsing models, such as those of Duchier and Debusmann (2001) and Penn and Haji-Abdolhosseini (2003), in that it represents syntax as two parse trees for the same sentence. Topological parsing sees a (freer-word-order) language as abiding by the constraints of two grammars: a tecto-grammar encoding constituent ordering rules and a pheno-grammar encoding restrictions on relations between constituents (including verb-argument structure and all morphological agreement constraints). The factored model differs from topological parsing by relying on very simple constraints between the trees, and by its heavy dependence on statistical knowledge. Moreover Klein and Manning mix phrase structure and dependency structure, unlike the topological models, which commit entirely to either dependency grammar (Duchier and Debusmann, 2001) or phrase struc-

[2] In this representation, which will be used in figures throughout the document, dependency structure is represented in the usual way and constituency is represented by nested 'bubbles'. Constituents with solid outlines have their contents *hidden* from the rest of the tree, except immediate siblings. Constituents with dashed outlines have their contents *exposed* to the rest of the tree, up to their closest hiding ancestor. For the moment, the reader can think of *exposed* constituents as *head* constituents.

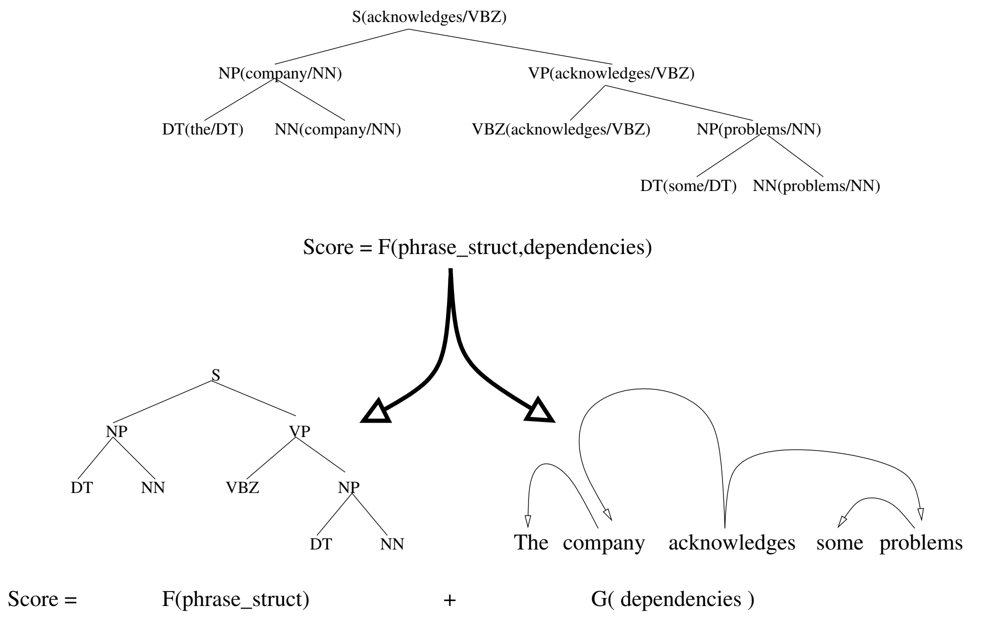

Figure 2.2: The factored language model analysis of the sentence “the company acknowledges some problems”, from the Penn treebank II (Marcus et al., 1994).

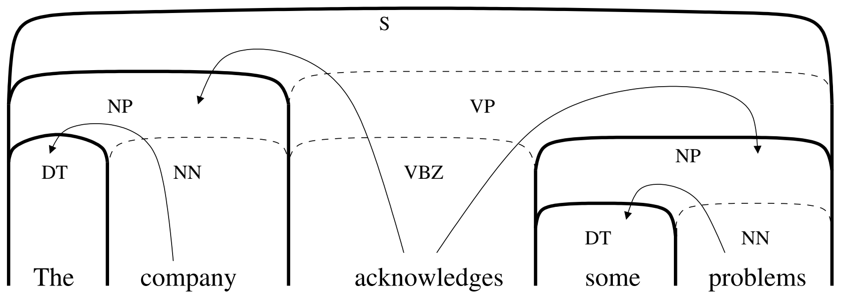

Figure 2.3: Constraints between tree representations in a factored model, in an example built from the sentence “the company acknowledges some problems”, from the Penn treebank II (Marcus et al., 1994).

ture grammar (Penn and Haji-Abdolhosseini, 2003). The purpose of the model differs as well: Klein and Manning aim primarily to produce a modular statistical model, while the topological schemes are designed to leverage known word order regularities to assist parsing in freer-word-order languages, something completely beyond the scope of the work by Klein and Manning (2002). Nonetheless, links between factored statistical parsing and topological parsing, in particular as envisioned by Penn and Haji-Abdolhosseini (2003), are a primary inspiration for this thesis.

There is also some resemblance between the factored model and synchronous grammar (early variants by Shieber and Schabes, 1990; Wu, 1997; Alshawi et al., 2000). A synchronous grammar is actually two grammars, that generate two independent sentences (or, as originally proposed by Shieber and Schabes, 1990, a sentence and a semantic representation thereof). The grammars are synchronized by linking production rules (or elementary trees in a tree-adjunction or tree-substitution grammar): each derivation step by one grammar entails a specific corresponding derivation step by the other grammar. In a companion to the proposal of Shieber and Schabes (1990), Abeillé et al. (1990) demonstrate the promise of synchronous grammar for machine translation. This application of the framework has recently become both fashionable and successful, especially for languages of freer-word-order (Eisner, 2003; Chiang, 2005; Ding and Palmer, 2005, for instance). The model of Klein and Manning (2002) can be seen as a synchronous grammar generating two different parse trees constrained to have the same terminal yield. This will not be very helpful in the present study, however, because we will begin by relaxing the constraints between phrase structure and dependency structure, and in doing so we will lose any plausible one-to-one correspondence between rules in both grammars.

2.2.4 Deriving Other Annotations from Context-Free Trees

A variety of approaches have been used to recover some of the 'deeper' annotations of Penn treebank-style corpora. The Collins model 3 (revisited in Collins, 2003), for instance, takes advantage of the history-based architecture of his models 1 and 2 to express WH-movement as an element of a generative language model: whenever a WH-constituent is generated in the language model, its generation is encoded in the history of future decisions as a 'gap' which at some point must be discharged as an empty 'trace' constituent. Collins argues that the handling of movement properly belongs in the core parsing model, and not in a subsequent annotation stage. He is concerned that movement impacts predicate subcategorization modeling (which is of value both in terms of improving accuracy and in terms of providing a basis for semantic analysis.)

Dienes and Dubey (2003) follow Collins in his approach to recovering information about traces: they use the same 'gap'/discharge mechanism in their language model, but they help the parser by explicitly marking the position of empty 'trace' constituents in the parser's input. This pre-processing step is accomplished by a *trace-tagger* trained to predict the position of empty constituents in raw (unparsed) text. By using a pre-processor, Dienes and Dubey (2003) abandon the idea of a single unified model, advocated by Collins. Competitive with these systems are approaches such as those of Johnson (2001) and Levy and Manning (2004) who prefer to recover movement annotations (and other 'deep' annotations involving empty constituents) in a post-processing step after a full parsing process. The present thesis can be seen as siding with Collins, preferring that as much analysis as possible be done by a single language model, thus permitting all grammatical elements to be weighed against one another rather than giving some elements (local relations) priority over others (long-range relations.)

2.2.5 Probabilistic Phrase Structure Parsing of German

While the nested phrase structure representation is seemingly appropriate for the analysis of English syntax, challenges arise when dealing with languages of freer-word-order, such as German. In German, two issues in particular need to be highlighted:

1. constituents may be broken up: in particular arguments or modifiers of a head

word may be dislocated from the rest of the phrase; and

2. the ordering of verbal arguments is far less predictive of their grammatical roles than it is in English.

In the German statistical phrase structure parsing community, annotators of gold-standard phrase structure treebanks such as NEGRA (Skut et al., 1997) and TüBa-D/Z (Telljohann et al., 2005), in an effort to stay theory-neutral, abandon efforts to represent all syntactic relations with deeply nested constituents. Phrase structure trees are flattened, reducing (but not always eliminating) constituent discontinuities, and constituents are labeled with detailed grammatical roles to compensate for information lost by the elimination of deep nesting.

Phrase structure parsing of German had a rocky start. Fissaha et al. (2003), Dubey and Keller (2003), Dubey (2005) and Schiehlen (2004) try to apply PCFG-based parsers to the NEGRA treebank, and are not able to achieve results comparable to those on English corpora.

Fissaha et al. (2003) use a simple unlexicalized PCFG model on the NEGRA treebank, providing correct part of speech tags as input. They observe the effects of varying the amount of training data (increasing coverage, but decreasing accuracy until coverage exceeds 85%), of labeling constituents with grammatical function (modest improvement in coverage and accuracy), and of parent encoding (modest improvement in accuracy, but significant penalty to coverage.) Their goal is to explore the parameter-space, rather than to provide an experimental baseline, but their results on development data are similar to those of other studies of statistical NEGRA parsing.

Dubey and Keller (2003) begin by comparing a reimplementation of Collins (1997) with an unlexicalized and a lexicalized PCFG parser, both realized by the LoPar system of Schmid (2000) (the lexicalized model is based on Carroll and Rooth, 1998). They find that the lexicalized PCFG model and the Collins model do not achieve constituent precision and recall superior to that of the unlexicalized parser. It may be worth noting that the Collins model *does* have quite substantially the best crossing bracket scores, although this anomaly is not explored. Dubey and Keller (2003) set aside the idea of conditioning children on their parent's lexical head, but do not abandon lexicalization entirely: they report their best results from a parser that conditions each child on the head of its immediately preceding sibling. This best result is no better than 71% labeled constituent precision and recall, up to 74% with perfect part of speech input.

Dubey (2005) ignores lexicalization entirely, and focuses on unlexicalized PCFG grammar (with Markov rules.) Probability smoothing combined with grammatical function labeling, supported by suffix analysis of unknown and uncommon words in pre-terminal rules, brings labeled constituent F-score up to 76%. Error analysis indicates that perfect part of speech tagging (including grammatical function!) brings this up to 85%.

Schiehlen (2004) reproduces the experiments of Klein and Manning (2003) on NEGRA, but finds that the parent encoding and Markovization used by Klein and Manning (2003), while improving constituent precision and recall, are actually detrimental to recovering syntactic dependency structure in NEGRA. He argues that excessive attention to English phrase structure might have led to models compromised in their ability to recover dependencies in other languages. His highest labeled constituent F-score of 71% is similar to that of Dubey and Keller (2003). His highest labeled dependency F-score of 82% is produced by a parser that does not achieve his best constituent scores. This parser relies on a number of detailed corpus-specific grammar re-writes.

Many of these negative results in German are called into question by Kübler (2005) and Kübler et al. (2006), who are able to achieve much better accuracies using the Stanford Parser (Klein and Manning, 2002, 2003) on the TüBa-D/Z treebank (Telljohann et al., 2005). These authors specifically examine the differences in performance between NEGRA and TüBa-D/Z , using the same parsing model, and find TüBa-D/Z easier to parse both in terms of phrase structure and in terms of verb-argument structure. While the Stanford Parser only achieves a labeled constituent F-score of 70% on NEGRA, it

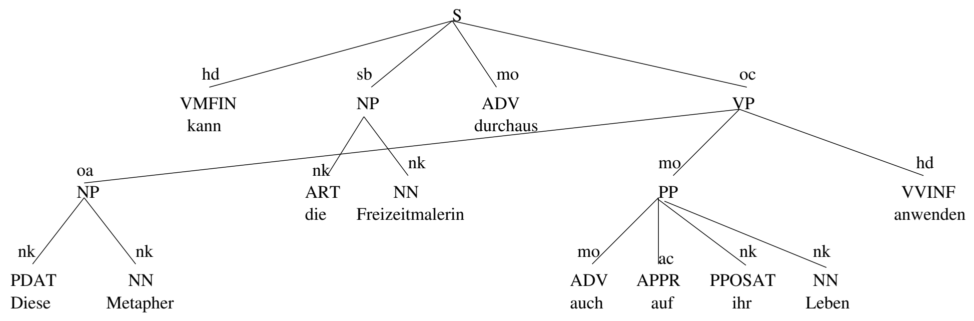

Figure 2.4: The NEGRA treebank analysis of an example from Kübler et al. (2006), graphically reformatted.

achieves a much more impressive 89% on TüBa-D/Z . They conclude that the improved results are because of differences in the annotation scheme (both treebanks were derived from newspaper text.) The NEGRA corpus groups constituents together by their roles in verb-argument structure, permits discontinuous phrases, and does not specify much of the structure internal to noun and prepositional phrases (figure 2.4.) By contrast, TüBa-D/Z groups constituents in topological fields defined by the positions of finite and non-finite verbs, *does not* use discontinuous phrases[3] and *does* specify structure internal to noun and prepositional phrases (figure 2.5.) Nonetheless, even on the TüBa-D/Z corpus, verb argument structure and noun-phrase modifiers cannot be recovered as reliably as they can in English (Kübler et al., 2006; Kübler and Prokić, 2006).

2.3 Statistical Dependency Parsing

Arguably, the problems encountered in German could be caused by flaws in the phrase structure representation of syntax, flaws that are not inherent to the older tradition of dependency representation (this argument is made, for instance, by Mel'čuk, 1988).

[3]Although the encoding of TüBa-D/Z trees does not explicitly represent discontinuous phrases, they can be reconstructed using edge labels and, where necessary, secondary links provided with the annotation.

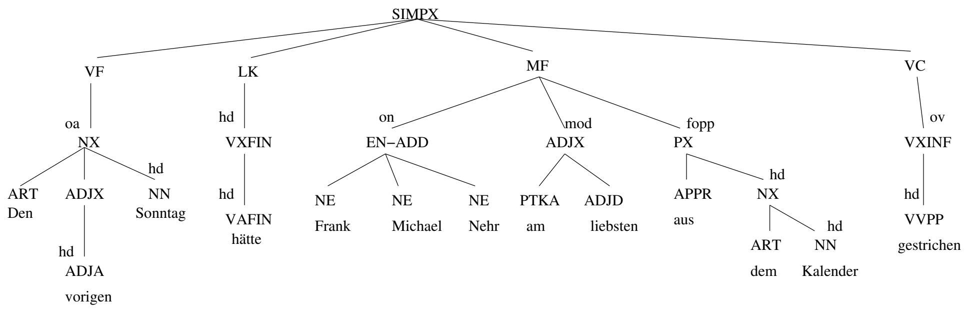

Figure 2.5: The TüBa-D/Z treebank analysis of an example from Kübler et al. (2006), graphically reformatted.

Dependency trees are labeled directed trees over the tokens of a sentence: the specific role of a token is encoded by the governing token assigned to it and by the syntactic label assigned to that governance relationship (figure 2.6 shows an unlabeled[4] dependency tree.) Critically, a dependency tree can be interpreted without reference to the relative order of tokens. It was unclear for some time, however, how dependency trees could be automatically extracted from sentences efficiently and accurately. In discrete parsing, general purpose constraint solvers can be bent to the task of parsing (see for instance Duchier, 1999; Duchier and Debusmann, 2001). In statistical parsing, however, very few constraints are employed, and most designers of parsing systems opt to implement their own search algorithms (some of which are discussed below), that are specific to the scoring model of their parsers.

2.3.1 Dependencies from Phrase Structure

Most of the work discussed above in section 2.2 treats only phrase structure analysis. The work of Levy and Manning (2004) is an exception, in that they evaluate their algorithm according to how well it is able to provide the basis for a dependency analysis (with

[4]Without labeling, dependency trees can fail to rule-out many syntactic ambiguities. Despite this, researchers (McDonald et al., 2005a; Yamada and Matsumoto, 2003; Eisner, 1996b) have often focused on the problem of finding governors while ignoring labeling. In this thesis, we will consider only unlabeled trees.

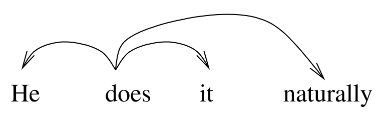

Figure 2.6: Unlabeled dependency analysis of an example adapted from Mel'čuk (1988), ambiguously representing the syntax of either "he does it naturally" or "he does it, naturally".

respect to both the English PTB and the German NEGRA corpora.) Schiehlen (2004) also focuses on extracting dependency trees from phrase structure analyses (and does so on NEGRA data), but argues that the statistical models he uses are biased toward producing good results on phrase based metrics and are not optimal with respect to dependency-based metrics.

Perhaps the most extreme example of using phrase structure analysis to recover dependency trees is provided by Collins et al. (1999). The authors parsed sentences from the Prague Dependency Treebank using a modified version of the Collins model 1 parser (Collins, 1997). They mapped the lexicalized phrase structure output of this parser to dependency trees, by assigning the head word of each child constituent as a dependent of the head word of its parent constituent. Training the model was less straight-forward: the training data were in dependency format, and lacked phrase structure. It was therefore necessary to use heuristic rules to map the dependency treebank to a phrase structure treebank as a preprocessing step before training.

2.3.2 Directly Finding Dependencies

Efforts to extract dependency trees from phrase structure trees are somewhat surprising in light of early work done by Eisner (1996b) and Eisner (1996a) to directly build dependency parsers from dependency-based probabilistic language models. In addition to providing an algorithm for efficiently computing Viterbi dependency parses (projective only), he also reports results very competitive with a state of the art phrase based parser of the time. But the idea seems to have persisted that statistical dependency parsing cannot be done accurately without the help of a phrase structure parser: Yamada and Matsumoto (2003), for instance, repeatedly point out that their dependency parser cannot be as accurate as one using a phrase based model, because it lacks information about phrase structure. Likewise, McDonald et al. (2005a) state: "It is well known that dependency trees extracted from lexicalized phrase structure parsers typically are more accurate than those produced by pure dependency parsers."

Arguably this belief in the superiority of phrase structure parsers over dependency parsers arose out of the fact that much more work had been done on statistical phrase structure parsers. A state-of-the art dependency parser (McDonald and Pereira, 2006) can now beat phrase structure parsers at producing dependency trees for some datasets. Given recent advances in statistical dependency modeling, more such results can be expected (consider for instance submissions to the CoNLL-X Shared Task on Multilingual Dependency Parsing, Buchholz and Marsi, 2006).

Two dependency parsers in particular deserve our attention. The first (Nivre, 2003; Nivre et al., 2004, 2006; Nilsson et al., 2006) is a deterministic shift/reduce-style parser, that scores trees based on the cost of their left-to-right derivations. The parser is interesting to us because of its approach to handling non-projective dependencies (sometimes referred to as crossing dependencies.) *Lifting* (Kahane et al., 1998) is the process of eliminating non-projectivity in dependency trees by assigning new governors to some tokens. If a token is chosen to be lifted, then its new governor (called *linear governor*) must be an ancestor of its original governor. What Nivre et al. (2006) do is to convert each tree in their training corpus of non-projective trees into a projective tree by lifting. They leave lift-path information in the converted trees to allow for recovery of the original trees, and train their parser to find projective trees *with associated lift traces.* After finding such a tree, their parser is then able to reconstruct the correct non-projective parse, by

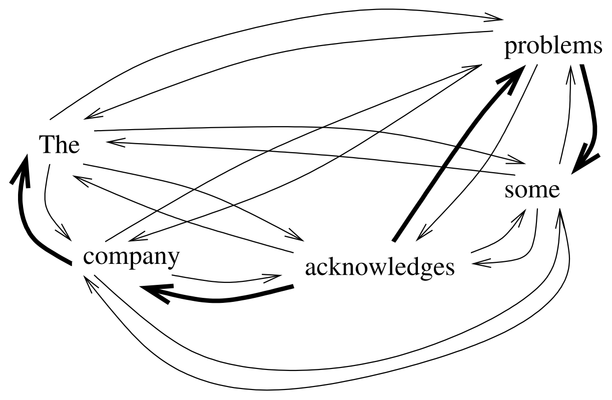

Figure 2.7: Maximum spanning tree parsing of the sentence "the company acknowledges some problems", from the Penn treebank II (Marcus et al., 1994) (the weights of all edges have been omitted from this diagram.)

un-lifting some children. There is a strong similarity between the idea of lifting and the ideas we will develop in this thesis. In particular, our parser effectively licenses some lifts, but not others, using a phrase structure grammar (the lifts are not explicitly performed by our parser, however.)

The second dependency parser of interest to us (McDonald et al., 2005a,b; McDonald and Pereira, 2006) uses an explicit language model. Early versions of the system exploit what is (to a computer scientist) one of the most elegant intuitions ever used in a statistical parser: given a sentence, the system scores every possible dependency edge between any two tokens (this is the contribution of the language model) and simply returns the maximum-spanning-tree (MST) of the complete graph of potential edges (this is the parsing algorithm.) This model is depicted in figure 2.7. The system inherently handles non-projectivity *without having to treat it as a special case.* McDonald and Pereira (2006) abandon the earlier MST parsing model in favour of an approximate method: trading the elegance of MST parsing for greater flexibility in the language model.

model	scope	interpretation	feature interdependence	training
generative probability	local decision	probability	feature interdependence must be explicitly managed by model designer	the ratio $p(x,c)/p(c)$ computed for each decision type
local max-entropy	local decision	probability	feature interdependence can be exploited by training algorithm	iterative-scaling or related method
random fields	up to entire solution	probability	feature interdependence can be exploited by training algorithm	iterative-scaling or related method, involves expensive normalization
large-margin	up to entire solution	artificial score	feature interdependence can be exploited by training algorithm	best-fit of boundaries between different values of x, or perceptron-based

Table 2.1: Comparison of some common statistical modeling techniques.

2.4 Statistical Pattern Recognition Techniques in Parsing

I now review some of the 'statistics' behind statistical parsing. This section is not meant to explore in depth all of the techniques used, and I do not claim to necessarily even be touching on the most important works relating to the topic. In these few paragraphs I aim to highlight the relationships between a number of common techniques (see the summary in table 2.1) and to note that the *grammatical formalism* adopted by a parsing system can be independent of the *mathematical modeling approach* used by the system. For instance, generative probability models have been used for both phrase structure parsers (e.g. Collins, 1999) and dependency parsers (e.g. Eisner, 1996b), as have perceptron-style models (e.g. Collins and Roark, 2004; McDonald et al., 2005a) and maximum-margin models (Taskar et al., 2004; Yamada and Matsumoto, 2003).

The tried and tested method (Collins, 1999, 2003; Eisner, 1996b; Charniak, 1996, 1997, 2000)[5] for scoring parse trees is a generative probabilistic model: each parsing decision (for a parsing model) or generation decision (for a language model) is assigned a probability conditioned on some limited context. Probabilities are estimated using ratios of feature co-occurrence frequencies extracted from a training corpus. The probability of a parse tree is the product of the probabilities of all the decisions that would have to be made to produce that tree. The advantages of this model are that it is intuitive and that training is fast. Modeling is limiting however: as new features are added, the model must be manually re-arranged. Furthermore, data-sparsity can rapidly become crippling as new interdependent features are added to a model based on frequency ratios.

An alternative approach to parsing uses maximum entropy modeling (Ratnaparkhi, 1997). Individual decisions are still assigned probability scores, but these probabilities are estimated not by taking ratios of frequencies, but by maximizing the entropy of a log-linear probability model, subject to the constraint that the model must exactly predict the frequency of all model features over the training corpus (for a more detailed mathematical description see Ratnaparkhi, 1998.) The advantage of this approach is flexibility: a relatively large number of features can be used to condition each decision, and the designer can add or remove model features without worrying about whether or not they are statistically independent. Training the model, however, can be rather time-consuming.

While Ratnaparkhi (1997) uses maximum-entropy estimates of probabilities for individual parsing decisions, Abney (1997) proposes using a maximum-entropy estimate of the probability of entire parse trees. Abney is particularly concerned with attribute grammars, where decomposing probabilities over full parse trees into probabilities over local decisions is difficult, because information about decisions relating to each attribute must be propagated as conditioning context to all other decisions constraining or constrained by the attribute. The proposed modeling framework, called *random fields*, while taking account of features relevant to individual decisions, does not estimate a normalized distribution for *each* decision, but rather normalizes only over complete solutions. Lafferty et al. (2001) provide an interesting discussion of the advantages of this approach in dealing with sequences of decisions. Exactly training such models for parsing is, however, infeasible, because of the need to normalize over all possible complete parse

[5]This last work, despite its title (*A maximum-entropy-inspired parser*) and opening discussion on the advantages of maximum entropy models, describes a model that relies on ratios of frequencies for parameter estimation.

trees. Abney (1997) addresses this challenge by resorting to random sampling. Johnson et al. (1999) point out that it isn't necessary to normalize over *all possible* parse trees to get reasonable results: it is far more practical to normalize over all parse trees of those sentences actually in the training set. This remains prohibitive for highly ambiguous grammars.

To find a model that is capable of handling large numbers of mutually dependent features, that is capable of handling sequences of mutually dependent decisions, *and* that is feasibly trainable, some researchers have adopted methods that do not produce probability models (in other words, the scores of all possible solutions do not sum to 1.) Arithmetically, these look a lot like log-linear probability models, except that they are normalized differently, if at all. They derive their predictive power not from accurately approximating a probability distribution, but from establishing a boundary (generally a hyperplane) in the model's feature space between correct structures (e.g. the desired parse) and incorrect structures (all other parses.)

The language model used by McDonald et al. (2005a) is an excellent example of this approach. Merging the Margin Infused Relaxed Algorithm (MIRA) of Crammer and Singer (2003) and Crammer et al. (2004) with the Averaged Perceptron concept, introduced to natural language processing by Collins (2002) and Collins and Roark (2004), they are able to very quickly train a linear model over a huge number of varied and partially redundant features, resulting in state-of-the-art parsing accuracy. A notable aspect of their work is that, although their algorithm aims to discriminate between correct parses and incorrect parses by as large a margin as possible, it does not go to great lengths to enforce the margin between each example's correct solution and *all possible alternative solutions* (this can be compared against Taskar et al., 2004), but rather enforces the margin only against the best-k alternative solutions. A practical consequence of this method is that large portions of the search space can be completely ignored during parameter estimation. McDonald and Pereira (2006) are not the only ones advocating this direction; they celebrate the kinship of their work with the Learning as Search Optimization paradigm of Daumé and Marcu (2005).

2.5 Summary

We have reviewed some of the major elements of statistical parsing, focusing in turn on phrase structure parsing, on dependency parsing and on statistical modeling. I now wish to highlight some aspects of this review.

Firstly, the lexicalized probabilistic context-free parsers of the late 1990's have not been substantially surpassed by newer systems in the task of English phrase structure parsing. This may speak to the need for better parsing benchmarks (for instance evaluation of parsers according to their suitability for integration into larger applications), but it can also be seen as evidence for the (relative) strength of the lexicalized PCFG model.

These parsers have not, however, fared as well in the domain of German parsing. Part of the problem here is a lack of clarity about what German phrase structure analysis should look like, and to what extent it is even a good idea. On the other hand, dependency parsers have been showing better and better results, and the dependency representation of syntax is at a particular advantage in the analysis of freer-word-order languages such as German.

It is in this light that I would like to point out an as-yet-unexploited opportunity provided by Klein and Manning (2002). Their idea of building a factored parser that simultaneously performs both phrase structure parsing and dependency parsing, using *separate language models* for each, may allow systems to be built that can combine a constituent analysis of phrases with a robustness to freer-word order. This view is the starting point of my thesis.

Finally, it is interesting to note the huge variety of statistical tools that have been brought to bear on statistical treebank parsing. If anything, the feverish exploration of techniques has intensified in the last five years, and it is to be hoped that these explorations will result in exciting new developments in parsing and in natural language processing generally.

Chapter 3

Context-Free Filtering Grammar

3.1 Overview

The factored parsing model of Klein and Manning (2002) provides an interesting framework in which phrase structure and dependency parsing can be performed in an interleaved manner. Unfortunately, the framework as initially presented by Klein and Manning (2002) does not permit the use of linguistically plausible dependency trees: the dependency trees are constrained to match the phrase structure very closely, one consequence of which is that the dependency trees must be projective.

In this chapter I present an extension to their model based on a new grammar formalism, *context-free filtering grammar*. The new style of grammar provides more flexibility than the original by permitting more than one token of a constituent to behave like a head (in particular, to govern any token within the constituent and to be dependent on tokens outside the constituent.)

After a discussion of the new grammar formalism, I present a chart-parsing algorithm and a statistical model to score the algorithm's output. The model uses PCFG to score phrase structure trees and the linear model of McDonald et al. (2005a) to score dependency trees; combined trees are scored by summing the log PCFG score and the dependency score.

Like Klein and Manning (2002), I use an A* search to find the best combined parse under the model. A* requires an admissible and monotonic scoring heuristic to predict the optimal score of any full solution incorporating any particular partial result. I use outside scores to build a heuristic for the PCFG model and a maximum-weight incident edge heuristic for the dependency model.

3.2 Grammar for Factored Model Parsing

I begin this section by defining the new grammar formalism: context-free filtering grammar. The formalism is inspired by the factored parsing model of Klein and Manning (2002), and I show that the formalism implied by their work is a special case of context-free filtering grammar. I discuss the representational advantages of the new formalism, its potential for use in dynamic programming algorithms and an extension of the formalism to handle dependency structures that are not fully connected.

3.2.1 Context-Free Filtering Grammar

Informally, a context-free filtering grammar defines a set of syntactic structures, each of which consists of a dependency tree and a phrase structure tree. The grammar defines the allowable phrase structures, and the dependency trees that can co-occur with each phrase structure. Phrase structures are licensed in roughly the same way that they are by a CFG, with the addition that the grammar must mark each child of each constituent with an exposure, either *Exposed* or *Hidden*. At least one child per constituent must be marked *Exposed*.

In a corresponding dependency tree, tokens contained in *Exposed* children may act as governors to tokens of sibling constituents and may be dependents of tokens outside the parent constituent. Tokens contained in *Hidden* children may act as governors of tokens only within the same child and may be dependents only of tokens within the parent constituent. As we will see in section 3.2.3, the label *Exposed* can be thought of as a generalization of the label *Head* from more traditional lexicalized phrase structure formalisms.

In devising this framework, I found it useful to think of dependency trees as ‘skeletons’ and phrase structure trees as ‘shells’. The phrase structure trees ‘contain’ the dependency trees, and restrict possible attachments.

Formally, a context-free filtering grammar G is a 4-tuple $\langle \mathcal{N}, \mathcal{T}, \mathcal{P}, S \rangle$, where:

- $\mathcal{N}$ is a set of *non-terminal symbols*;
- $\mathcal{T}$ is a set of *pre-terminal symbols* (parts of speech);
- $\mathcal{P}$ is a set of *production rules*;
- $S \in \mathcal{N}$ is the *start symbol.*

Each production rule in $\mathcal{P}$ has the form:

$$L \leftarrow R_0^{e_0} R_1^{e_1} ... R_n^{e_n}$$

where $L \in \mathcal{N}$, $R_i \in \mathcal{N} \cup \mathcal{T}$ and $e_i \in \{Hidden, Exposed\}, \forall i$ from $0..n$. At least one of $e_0, e_1, ..., e_n$ must have the value *Exposed.*

Informally, $(L \leftarrow R_0^{e_0} R_1^{e_1} ... R_n^{e_n})$ is the same as a context–free grammar production $(L \leftarrow R_0 R_1 ... R_n)$, except that each right–hand–side symbol is marked as either exposing or hiding its tokens.

An example of a context–free filtering grammar is shown in figure 3.1. This simple grammar will be used in most of the illustrative examples of this chapter. A larger grammar fragment for German (sufficient to generate the example in figure 3.10) is shown in figure 3.2, and will be used for some motivational examples.

Since G is essentially a context-free grammar, we will adopt the familiar notation for phrase structure derivations: we can speak of a derivation $(D : S \rightarrow_G^* \sigma)$ of the string

$$\begin{array}{rcl}
\mathcal{T} & = & \{C\} \\
\mathcal{N} & = & \{A, B\} \\
S & = & A \\
\mathcal{P} & = & \{ \\
& & \quad A \leftarrow B^{Exposed} B^{Exposed}, \\
& & \quad A \leftarrow B^{Exposed} A^{Hidden}, \\
& & \quad B \leftarrow B^{Hidden} B^{Exposed}, \\
& & \quad B \leftarrow C \\
& & \}
\end{array}$$

Figure 3.1: Sample context-free filtering grammar. The pre-terminal C will expand to terminals a, b, c.

σ from S according to G. If we restrict ourselves to considering left-most depth-first derivations, then we can also speak of the (labeled) phrase structure tree P induced by D.

To formally describe the constraints imposed by the phrase structure on dependency-structure, we will need some notation to discuss relationships between constituents of a phrase structure tree P. If constituent C_1 is contained within C_2 (and $C_1 \neq C_2$), we will write that C_2 *strictly dominates* C_1 ($C_2 \rhd C_1$). In an abuse of notation, we will say that a pre-terminal constituent dominates its matching terminal, despite the fact that G (in its role as a CFG) only generates the pre-terminals.

If two constituents C_1 and C_2 are in P, then we will refer to their closest common ancestor C^* as the *join* of C_1 and C_2 ($C^* = C_1 \sqcup_P C_2$).

If p is a phrase structure tree with terminal x in its yeild, then let $envelope_p(x)$ be the smallest $Hidden$ constituent in p that dominates x, or the root constituent if x is not contained in any $Hidden$ constituent.

We will represent dependency trees as *connected directed acyclic graphs*, in which each vertex is labeled with a terminal (word token), pre-terminal and linear position (from the string σ), and edges represent dependencies. As is traditional in work on dependency

$$
\begin{array}{rcl}
\mathcal{T} & = & \{NN, PRELS, PIDAT, VMFIN, VAFIN, ADV, PIS, VVINF, PPER, VVFIN\} \\
\mathcal{N} & = & \{NX, VF, SIMPX, MF, C, VXFIN, LK, ADVX, VXINF, R\_SIMPX, VC, NF\} \\
S & = & SIMPX \\
\mathcal{P} & = & \{ \\
& & SIMPX \leftarrow VF^{Hidden} LK^{Exposed} MF^{Exposed} VC^{Exposed} NF^{Hidden}, \\
& & R\_SIMPX \leftarrow C^{Hidden} MF^{Hidden} VC^{Exposed}, \\
& & C \leftarrow NX^{Exposed}, \\
& & VF \leftarrow NX^{Exposed}, \\
& & LK \leftarrow VXFIN^{Exposed}, \\
& & MF \leftarrow NX^{Exposed}, \\
& & MF \leftarrow ADVX^{Exposed} NX^{Exposed}, \\
& & MF \leftarrow NX^{Exposed} NX^{Exposed}, \\
& & VC \leftarrow VXFIN^{Exposed}, \\
& & VC \leftarrow VXINF^{Exposed}, \\
& & NF \leftarrow R\_SIMPX^{Exposed}, \\
& & NX \leftarrow NX^{Exposed} R\_SIMPX^{Hidden}, \\
& & NX \leftarrow NN^{Exposed}, \\
& & NX \leftarrow PRELS^{Exposed}, \\
& & NX \leftarrow PPER^{Exposed}, \\
& & NX \leftarrow PIDAT^{Hidden} NN^{Exposed}, \\
& & VXFIN \leftarrow VMFIN^{Exposed}, \\
& & VXFIN \leftarrow VAFIN^{Exposed}, \\
& & VXFIN \leftarrow VVFIN^{Exposed}, \\
& & VXINF \leftarrow VVINF^{Exposed}, \\
& & ADVX \leftarrow ADV^{Exposed} \\
& \} &
\end{array}
$$

Figure 3.2: Context-free filtering grammar fragment for German.

grammar, the edge will point from the governor to the dependent. One vertex, designated the root, has no incoming edges, all other vertices have exactly one incoming edge. In figures, the root will be denoted by an incoming edge from outside the graph. We will call all such trees *dependency trees of* σ.

Let σ be a string and let G be a context-free filtering grammar. Let p be the parse tree induced by a left-most depth-first derivation $D : S \rightarrow_G^* \sigma$, and let $g = \{\mathcal{V}, \mathcal{E}\}$ be a dependency tree of σ.

Then, together, p and g are a *parse of* σ if and only if:

1. $(u, v) \in \mathcal{E} \Rightarrow \mathit{envelope}_p(u) \triangleright v$ (this enforces the constraint that tokens in *Hidden* constituents may only be governors of tokens within the same constituent, by stating that dependency edges may not leave a constituent marked *Hidden*);

2. $(u, v) \in \mathcal{E} \Rightarrow \nexists i$ such that $u \sqcup_p v \triangleright i \triangleright \mathit{envelope}_p(v)$ (this enforces the constraint that a token in a *Hidden* constituent may only be a dependent of a token within its immediate parent constituent, by stating that dependency edges may not enter a constituent marked *Hidden* from anywhere other than within the parent constituent).

Figure 3.3 demonstrates condition 1 in an example, and figure 3.4 demonstrates condition 2.

3.2.2 Sample Derivation Using a Context-Free Filtering Grammar

There are a variety of ways to construct a syntactic structure according to a context-free filtering grammar. One can first derive a phrase structure, and then look for a compatible dependency tree; one can first choose a dependency tree, and then look for a compatible phrase structure; or one can assemble both structures at once, bottom up. Foreshadowing the parsing algorithm I will introduce, I provide an example of the last approach. To

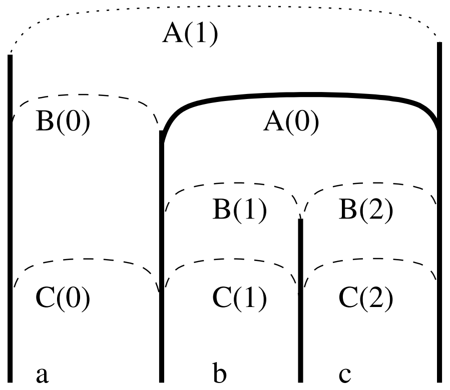

Figure 3.3: Demonstration of constraint 1: hidden tokens cannot govern.

In this example, the $A_{(0)}$ constituent is hidden and all others are either exposed or are the root. If we look at the terminal b, it may govern c in a dependency relation, but may not govern a, because a does not lie within b's envelope $A_{(0)}$.

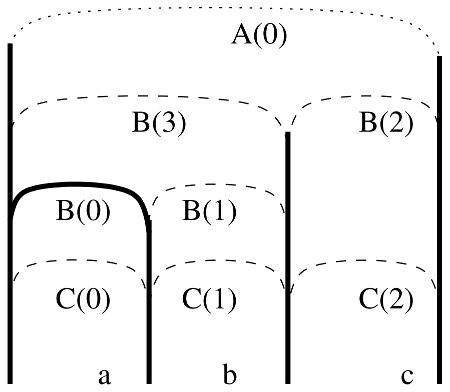

Figure 3.4: Demonstration of constraint 2: hidden tokens can only be seen from parent. In this example, the $B_{(0)}$ constituent is hidden and all others are either exposed or are the root. If we look at the terminal a, it may depend on b, but may not depend on c, because the nearest common ancestor of a and c is $A_{(0)}$, which strictly dominates the immediate parent of a's envelope.

emphasize the role of the constraints, I show all allowable dependency edges permitted by each constituent that is built, highlighting in bold the edges of one particular dependency tree as it is built, bottom-up.

The grammar used in this derivation is in figure 3.1. The derivation is shown in figure 3.5. Each 'chunk' of the diagram represents one passive chart arc in a chart parse. Only those chart arcs contributing directly to the final parse are included in the diagram. Upward blocky arrows indicate which chart arcs have been combined into each larger construction. Thin arrows within each chart arc indicate allowable dependency edges in that arc; thick arrows indicate committed dependency edges.

Attached subconstituents are enclosed either with a solid line (to indicate that their contents have been hidden by their parent) or a dashed line (to indicate that their contents have been exposed by their parent.) Unattached subconstituents are drawn with a dotted line: a constituent's exposure is only defined in relation to its parent constituent.

3.2.3 Factored Lexicalized PCFG as a Special Case

The context-free filtering grammar approach is best seen as a generalization of a simpler approach, the factored parsing model of Klein and Manning (2002).

Klein and Manning (2002) propose the factored model in an effort to disentangle the statistical models of phrase structure and of lexical headedness in lexicalized phrase structure parsers. They recommend simultaneously searching for a phrase structure tree and a compatible dependency tree, using an A* search. Because the starting point for their model is lexicalized PCFG parsing, their notions of 'dependency tree' and 'compatibility' are fairly limited. In particular:

- each phrase structure constituent must have exactly one head child, and the lexical head of the constituent is equal to the lexical head of that child; and

- the lexical heads of all other children of a constituent must be immediately governed

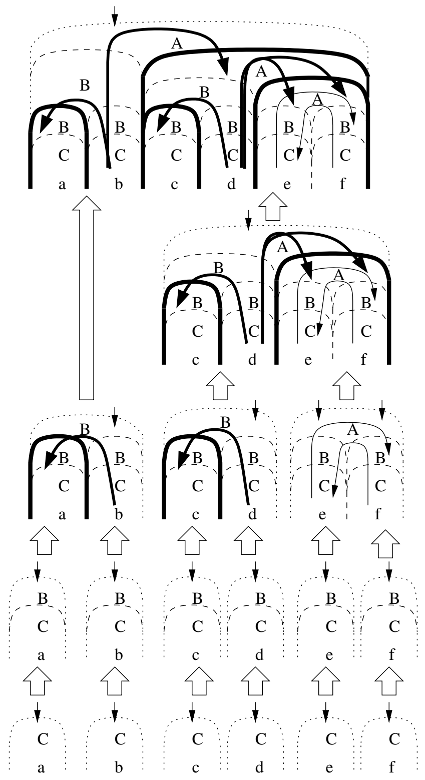

Figure 3.5: Example derivation of a string using a the toy grammar from figure 3.1

.

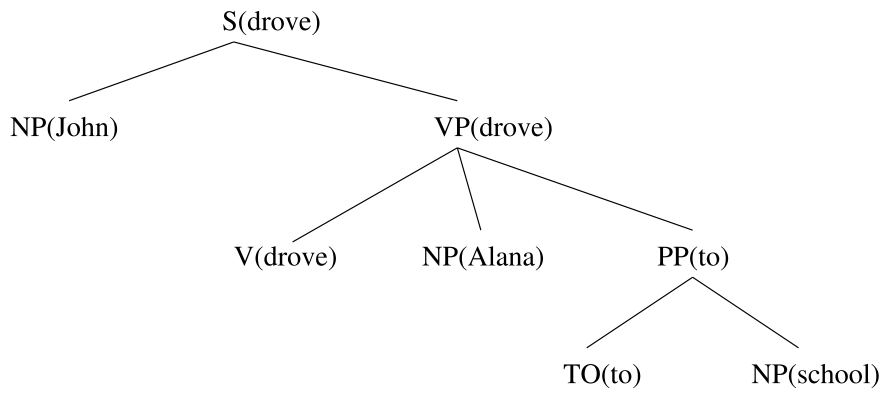

Figure 3.6: A simple lexicalized parse tree.

by the head child's lexical head.

Consider the following grammar, in which head symbols are marked with a *:

$$
\begin{aligned}
\mathcal{T} &= \{\mathrm{V}, \mathrm{NP}, \mathrm{TO}\} \\
\mathcal{N} &= \{\mathrm{S}, \mathrm{VP}, \mathrm{PP}\} \\
\mathcal{P} &= \{ \\
&\quad \mathrm{S} \leftarrow \mathrm{NP}\ \mathrm{VP}^* \\
&\quad \mathrm{VP} \leftarrow \mathrm{V}^*\ \mathrm{NP}\ \mathrm{PP} \\
&\quad \mathrm{PP} \leftarrow \mathrm{TO}^*\ \mathrm{NP} \\
&\}
\end{aligned}
$$

With this grammar, we can find a lexicalized parse tree for "John/NP drove/V Alana/NP To/TO School/NP" as shown in figure 3.6. We can trivially express this grammar (and any context-free grammar with one marked head child per rule) as a context-free filtering grammar, according to the following observations.

Recall that context-free filtering grammar marks at least one child of each constituent as *Exposed* (others may be marked *Hidden*) and imposes the following constraints on dependency structure:

1. an edge may only *leave* exposed constituents;

2. an edge may *enter* a hidden constituent if it is the first constituent that the edge enters;

The class of grammars of the factored model of Klein and Manning (2002) is precisely the subset of context-free filtering grammars in which *only one* child is exposed in each constituent. In particular:

- the one *exposed* child in each constituent is the *head* child of the factored model;
- only one token is exposed within each constituent (provable by induction on the height of the phrase structure); this token is the *lexical head* of the constituent under the factored model;
- since only one token is exposed within a constituent, that one token is the only candidate governor for the lexical heads of the other children, and it follows that the lexical heads of the other children will all be immediate dependents of that token (the head child's lexical head.)

The context-free filtering grammar version of the example grammar from above is:

$$\begin{aligned}
\mathcal{T} &= \{V, NP, TO\} \\
\mathcal{N} &= \{S, VP, PP\} \\
\mathcal{P} &= \{ \\
&\quad S \leftarrow NP^{Hidden} VP^{Exposed} \\
&\quad VP \leftarrow V^{Exposed} NP^{Hidden} PP^{Hidden} \\
&\quad PP \leftarrow TO^{Exposed} NP^{Hidden} \\
&\}
\end{aligned}$$

The corresponding parse tree for “John/NP drove/V Alana/NP To/TO School/NP” is shown in figure 3.7.

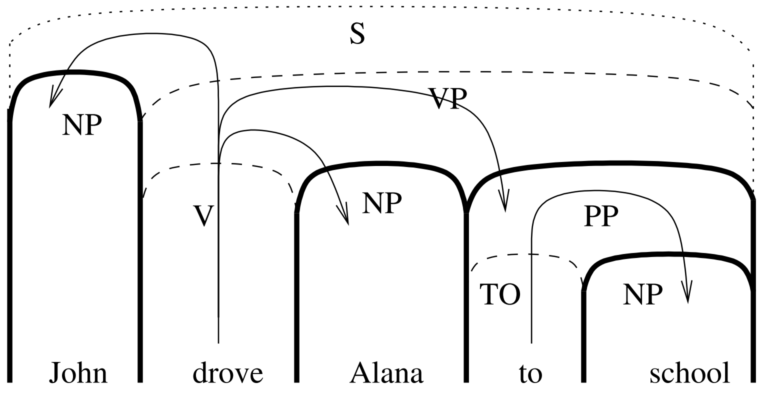

Figure 3.7: A simple lexicalized parse tree shown in a factored representation.

3.2.4 Properties of Context-Free Filtering Grammar

Additional Flexibility

Figure 3.8 provides an example of a phrase structure annotation that does not admit a linguistically plausible dependency structure (such as figure 3.9) under the factored model of Klein and Manning (2002).

A number of problems are evident.

- Topological fields don't have lexical heads. The Mittelfeld (`MF`) constituent in particular contains two tokens properly in dependency relations with tokens from other fields.

- Although the main clause can be said to have a head ('würden'), not all tokens exposed within the clause are properly dependents of it (e.g. 'jemanden' ought to be governed by 'finden'.)

- A good dependency tree for this sentence is not projective: in particular, the relative clause is governed by the token 'jemanden', and this dependency must cross the relation between 'würden' and 'finden'.

All of these problems can be overcome with a context-free filtering grammar. The parse

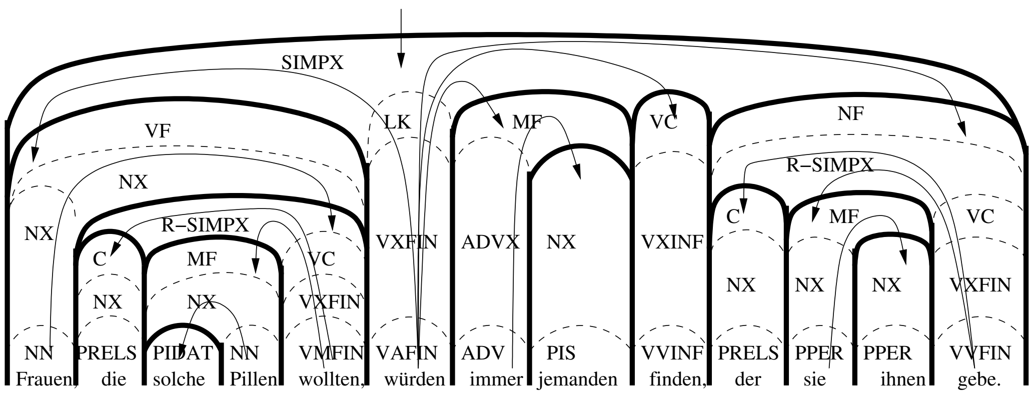

Figure 3.8: A phrase structure / dependency structure pair in the style of Klein and Manning (2002).

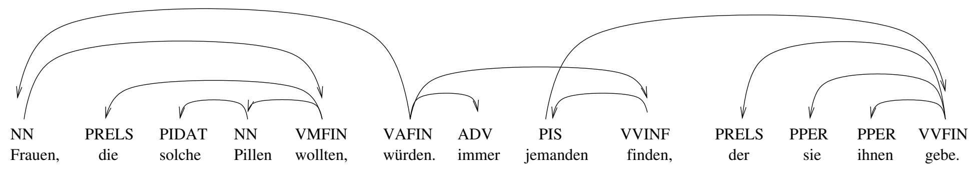

Figure 3.9: A better dependency tree.

tree shown in figure 3.10 is derived from the example grammar of figure 3.2; the derivation is shown in figure 3.11.

Hidden Structure

The additional flexibility of the new formalism comes at a cost of greater 'interconnectedness' within its parse trees. In a traditional lexicalized phrase structure tree, each constituent is characterized by a *phrase category* label and a *single exposed token.* Constituents with the same category and head are interchangeable syntactically. This interchangeability permits the development of efficient parsing algorithms based on dynamic programming. In a syntactic structure licensed by a grammar of the new formalism, each constituent is characterized by a phrase category label and a *set of exposed tokens.* Constituents are interchangeable syntactically only if their category and entire set of ex-

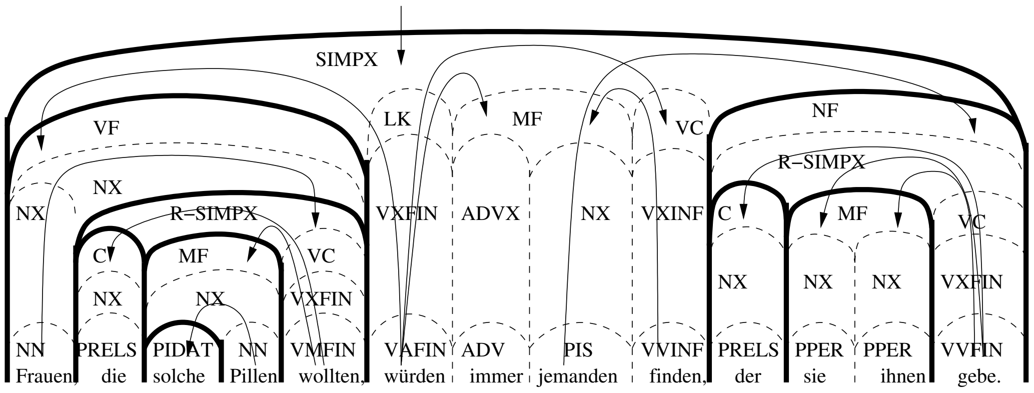

Figure 3.10: A phrase-filtered dependency tree.

posed tokens match. If a grammar permits the creation of constituents with arbitrarily large exposed sets, then dynamic programming can no longer be relied upon to produce efficient parsing algorithms.

Fortunately it is possible, within the framework, to limit this production of arbitrarily large exposed sets to a small group of phrase categories. A phrase of category L can expose more than one token only if either:

- a rule with L in its left-hand-side marks more than one right-hand-side symbol as exposed, or
- a rule with L in its left-hand-side marks as exposed a symbol of another category that can expose more than one token.

It is therefore possible to construct a grammar for which many phrase categories are as ‘well-behaved’ complexity-wise as they would have been under a stricter lexicalized grammar framework, but for which a small number of categories have the greater freedom required to license plausible linguistic structures.

In the case of the German TüBa-D/Z treebank that will be used for the experiment in chapter 4, it is the case that phrase constituents (such as noun-phrases) will only ever expose a single head-token, whereas some field constituents (notably the Mittelfeld) will

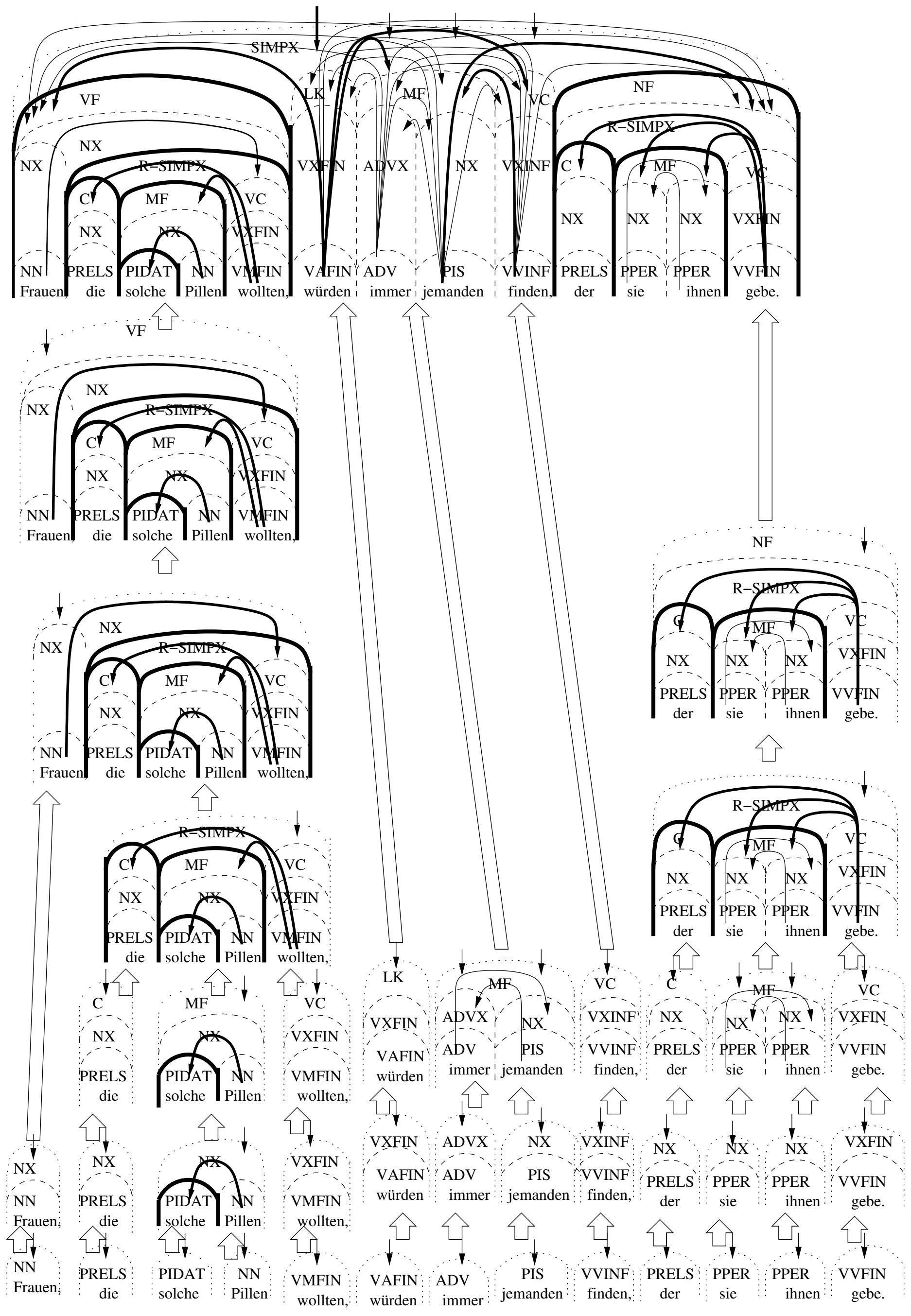

Figure 3.11: Example derivation of a string using a the toy grammar from figure 3.2.

need to expose multiple tokens. Clauses will also need to expose multiple tokens, to allow tokens from more than one field to act as governors within the clause, but the clauses themselves will normally be *Hidden* when embedded in other constituents, thus limiting the proliferation of exposed tokens.

Insufficiency for Discrete Analysis

The constraints imposed on dependency structure by context-free filtering grammar are not particularly discriminating: they serve to relate the dependency structure and the phrase structure to one another, but are not sufficient to rule out implausible dependency structures. A grammar formalism relying heavily on discrete logical constraints could benefit from using this formalism as a starting point, but would require additional constraints on dependency relations.

The formalism is adequate, however, for statistical parsing, in which the plausibility of syntactic structure is ultimately decided by the weight assigned to a structure according to a statistical model.

3.2.5 Handling Unconnected Dependency Trees

The experiment described later in this thesis is conducted on a modified version of the TüBa-D/Z treebank. The original corpus is a phrase structure treebank, in which some trees contain disconnected constituents. The phrase structure can be re-connected by means of simple heuristics, but it is unclear how to design simple and accurate heuristics for automatically creating a fully connected dependency tree from a disconnected phrase structure. In the derived corpus I will use, therefore, the phrase structure of an utterance is connected, but the dependency structure can be disconnected. In other words, there may be multiple root governors in the dependency structure. To permit these unconnected dependency structures, the context-free filtering grammar definition must be modified slightly.

$$\begin{array}{rcl}
\mathcal{T} & = & \{C\} \\
\mathcal{N} & = & \{A', A, B\} \\
S & = & A' \\
\mathcal{P} & = & \{ \\
& & \quad A' \leftarrow A^{Unlinked}, \\
& & \quad A \leftarrow B^{Exposed} B^{Exposed}, \\
& & \quad A \leftarrow B^{Exposed} A^{Hidden}, \\
& & \quad B \leftarrow B^{Hidden} B^{Exposed}, \\
& & \quad B \leftarrow C \\
& & \}
\end{array}$$

Figure 3.12: Sample context-free filtering grammar, modified from figure 3.1 to ensure that the start symbol always has *Unlinked* children.

In particular, we can extend the set of available exposures ($\{Exposed, Hidden\}$) with a third value, *Unlinked*. When a rule assigns the exposure *Unlinked* to a child, each token exposed within that child must either be governed by another token exposed within the child, or must be ungoverned (a root.) Tokens from an *Unlinked* constituent may not take part in dependency relations with tokens from sibling constituents.

To simplify parsing, we will want that *only* tokens of *Unlinked* constituents be allowed as roots. The easiest way to achieve this is to require that the start symbol S never expose any of its children. Instead of having at least one *Exposed* child, an S constituent must have at least one *Unlinked* child. Because it does not expose any tokens, the start symbol should never appear in the right-hand-side of a production rule. If this restriction poses a problem for a grammar, then the old start symbol S can be replaced by a new start symbol S', and the rule $S' \leftarrow S^{Unlinked}$ can be added to the grammar (example transformed grammar in figure 3.12.)

In the remainder of this thesis, context-free filtering grammar is to be understood as including this extension.

3.3 Parsing

How do we parse with a context-free filtering grammar? As with traditional lexicalized grammars we can use a bottom-up chart parser.

Passive arc signatures in this parser will consist of a phrase-label and a set of tokens exposed within the constituent spanned by the arc. Passive arcs will store both the internal phrase structure of the constituent, and all of the dependency relations involving hidden tokens spanned by the arc.

Active arcs will match lists of passive arcs to prefixes of production rule right-hand-sides. This matching will include an assignment of exposure to each passive arc.

3.3.1 Chart Arc Definitions

Strings can be parsed according to a context-free filtering grammar using a modified chart parser (chart parsing is sufficiently ubiquitous that introductions to the approach can be found in textbooks, such as Allen, 1994). We will assume that the input string is a sequence of terminal/pre-terminal pairs (i.e. another algorithm performs part of speech tagging beforehand.)

All chart parser arcs have the following components:

- *lhs_label*: the left-hand-side label of the matching production rule;
- *lpos*: the position of the first token covered by the arc;
- *rpos*: the position after the last token covered by the arc;
- *rhs*: the sequence of non-terminals (annotated with exposure) specified by the matching rule;
- *children*: the sequence of passive arcs matching the members of *rhs* (so far);
- *exposed_set*: the set of all positions exposed within this arc; and

- *inverse_dep_fun*: a partial function mapping positions in $\mathcal{D} = \{lpos, ..., rpos - 1\}$ to other positions in $\mathcal{D}$, this function is interpreted as mapping dependents to governors in a dependency structure, and the corresponding dependency structure must be acyclic.

The chart parser builds arcs of two forms.

- Active arcs have the additional components:
  - *rule_pos*: the index of the left-most member of *rhs* not matched by this arc (note: while *rpos* is an index into the sentence, *rule_pos* is an index into the current rule);
  - *hidden_sets*: the set of *exposed_sets* from the passive arcs in *children* marked *Hidden*;
  - *unlinked_sets*: the set of *exposed_sets* from the passive arcs in *children* marked *Unlinked*;

  In an active arc, *inverse_dep_fun* is defined for all elements of:

$$\begin{aligned}\mathcal{D}_A \quad = \quad & \{t \in \{lpos, ..., rpos - 1\} | \\ & \quad t \notin exposed\_set \\ & \quad \text{and} \quad \forall hs \in hidden\_sets, t \notin hs \\ & \quad \text{and} \quad \forall us \in unlinked\_sets, t \notin us \\ & \}\end{aligned}$$

  .

- Passive arcs have no additional components and, in a passive arc, *inverse_dep_fun* is defined for all elements of $\mathcal{D}_P = \{lpos, ..., rpos - 1\} \setminus exposed\_set$.

3.3.2 Parsing Actions

The modified chart parser behaves like a normal chart parser except in its assignment to arcs of *exposed_set*, *hidden_sets*, *unlinked_sets* and *inverse_dep_fun*, and in its generation of multiple passive arcs for each rule completion (one passive arc per allowable $inverse\_dep\_fun$.)

An active arc resulting from rule prediction (i.e. having no children) assigns the empty set to each of $exposed\_set, hidden\_sets, unlinked\_sets$ and an empty domain to $inverse\_dep\_fun$.

When an active arc a is formed from a smaller active arc a_a and an adjacent passive arc a_p, information from the two child arcs are incorporated into a, but no new decisions are made (i.e. no additional dependency relations are committed to.) More formally, the following assignments are made (the symbol '.' is used here to access components of arcs):

$$a.inverse\_dep\_fun(t) = \begin{cases} a_a.inverse\_dep\_fun(t) \\ \quad \text{if } t \in \text{dom}(a_a.inverse\_dep\_fun) \\ a_p.inverse\_dep\_fun(t) \\ \quad \text{if } t \in \text{dom}(a_p.inverse\_dep\_fun) \end{cases}$$

$$a.exposed\_set = \begin{cases} a_a.exposed\_set \\ \quad \text{if rhs symbol matching } a_p \text{ is not exposed} \\ a_a.exposed\_set \cup a_p.exposed\_set \\ \quad \text{if rhs symbol matching } a_p \text{ is exposed} \end{cases}$$

$$a.hidden\_sets = \begin{cases} a_a.hidden\_sets \\ \quad \text{if rhs symbol matching } a_p \text{ is not hidden} \\ a_a.hidden\_sets \cup \{a_p.exposed\_set\} \\ \quad \text{if rhs symbol matching } a_p \text{ is hidden} \end{cases}$$

$$a.unlinked\_sets = \begin{cases} a_a.unlinked\_sets \\ \quad \text{if rhs symbol matching } a_p \text{ is not unlinked} \\ a_a.unlinked\_sets \cup \{a_p.exposed\_set\} \\ \quad \text{if rhs symbol matching } a_p \text{ is unlinked} \end{cases}$$

Passive arcs can be generated in two ways: as a result of scanning one element of the input sequence, and as a result of passivizing an active edge with a fully matched rhs. Passive arcs generated by scanning have $exposed\_set = \{lpos\}$ and an $inverse\_dep\_fun$ defining no mappings, but are otherwise uninteresting.

When an active arc a_a has an element of $a_a.children$ matching each element of $a_a.rhs$, it can be passivized. Passivization involves generating a new passive arc p for each possible assignment of dependency relations with dependent tokens:

$$t \in \{a_a.lpos, ..., a_a.rpos - 1\} \setminus a_a.exposed\_set$$

.

Each new passive arc p inherits all of its component values from a_a, except for the $inverse\_dep\_fun$. The component $p.inverse\_dep\_fun$ may take on any mapping that satisfies the following constraints:

- The dependency graph represented by $p.inverse\_dep\_fun$ is acyclic;
- $a_a.inverse\_dep\_fun(t)$ is defined
  implies $p.inverse\_dep\_fun(t) = a_a.inverse\_dep\_fun(t)$
  (no dependencies assigned in children of a_a are altered);
- For each $t \in \{p.lpos, ..., p.rpos - 1\}$ that is also in some $hs \in a_a.hidden\_sets$,
  $p.inverse\_dep\_fun(t) \in hs \cup p.exposed_set$
  (tokens exposed within a hidden child must be dependent on tokens in the same child, or on tokens exposed within an exposed child);
- For each $t \in \{p.lpos, ..., p.rpos - 1\}$ that is also in some $us \in a_a.unlinked\_sets$,
  $p.inverse\_dep\_fun(t) \in us \cup \{\text{NO\_GOVERNOR}\}$
  (tokens exposed within an unlinked child must be dependent on tokens in the same child, or must be defined to have no governor);

A passive arc spanning the entire input and having a label matching the grammar's start symbol is a full parse of the input. Because a rule forming a constituent bearing

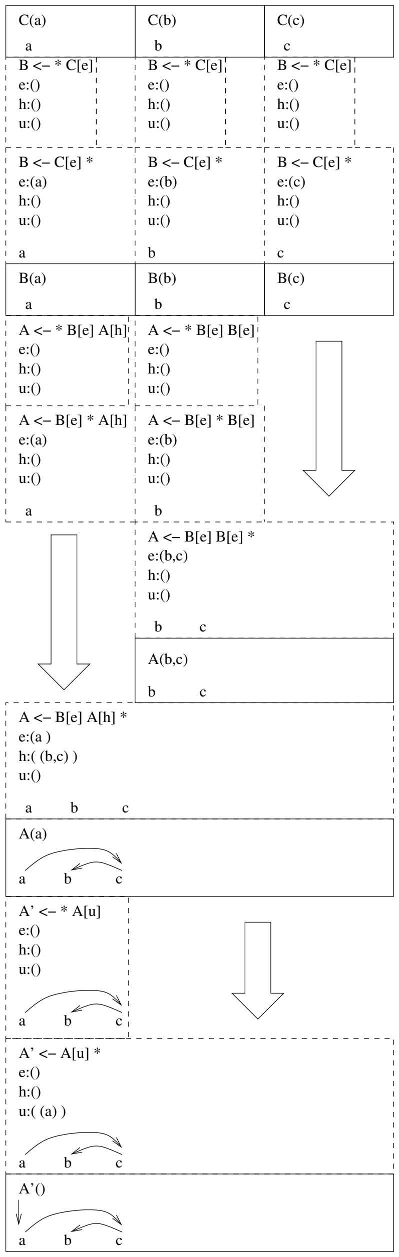

Figure 3.13: Chart parsing.

Passive arcs are in solid boxes, active arcs in dashed boxes. In active arcs, 'e:' denotes the exposed set, 'h:' the hidden sets and 'u:' the unlinked sets. Only chart arcs contributing to a single parse are shown.

the start label has no *exposed* children (recall the extension of section 3.2.5), we know that the entire dependency structure is assigned. A diagrammatic representation of a few steps of parsing with the grammar from figure 3.12 is given in figure 3.13.

Because of the inclusion of sets of tokens as components of arcs, it should be clear that the worst case time and space complexity of the modified chart parser will be exponential in the size of the input, even if we use a dynamic programming algorithm for greater efficiency. In my implementation, I aim to overcome this weakness by manipulating search order and by pruning.

3.4 Statistical Parametrization

I now turn to the question of how we can apply a statistical model to context-free filtering grammar. I address two questions at once: how to find the best parse according to the model (the decoding problem), and how to choose the model parameter values (the training problem.)

I begin by reviewing the approach to decoding using A* search, introduced by Klein and Manning (2002) for their factored model, which we also use with the new grammar formalism. The search requires an admissible heuristic for both bottom-up dependency parsing and bottom-up phrase structure parsing.

I describe the dependency scoring model, its heuristic and its training process. The dependency model that I use is a reimplementation of McDonald et al. (2005b). I follow with a description of the phrase structure model, its heuristic and its training process, based on the work of Klein and Manning (2003).

Finally, I describe a process for training the dependency model to take advantage of the information provided by the phrase structure model, leading to an anticipated improvement in the accuracy of the factored parser.

3.4.1 Factored Model Search

Following the work of Klein and Manning (2002), I propose a factored scoring model for context-free filtering grammar parses: a function g_p maps each possible phrase structure tree to a numerical score, another function g_d maps each possible dependency tree to a numerical score, and the total score of a pair of trees is composed from the values of g_p and g_d by a third function h. In our case h simply adds its two arguments together, but more complex arrangements are conceivable.

For this factored model to be of any use, we need a search method that will find the best combined score over all allowed pairs of phrase structure and dependency trees. As pointed out at the end of section 3.3, the search space of our chart parser is potentially exponential in size, so an exhaustive search is not feasible. Fortunately, chart parsers are quite flexible about the order in which they explore the space, and Klein and Manning (2002) have demonstrated that factored parsing models can be amenable to A* search.

A* search is a very general and well known algorithm used in artificial intelligence (and is therefore described in AI textbooks such as Russell and Norvig, 2003). Given any *admissible and monotonic scoring heuristic*, the A* search is guaranteed to find the model-optimal goal (a maximum in our case.) A scoring heuristic is a function that, given a partial solution S' (in our case, a chart arc) predicts the best possible score of a full solution derived from S'. A scoring heuristic is admissible if its prediction is never an underestimate, and monotonic if filling in more detail in a partial solution never leads to an increase in its predicted score. The amount of the search space that must be explored to find the solution is largely determined by the accuracy of this scoring heuristic.

Since our factored scoring model simply adds together g_p and g_d, we can design independent admissible and monotonic heuristics for both functions, and the sum of the heuristics will be an admissible and monotonic heuristic for the combined model.

3.4.2 Dependency-Structure Model

Linear Scoring Model

McDonald et al. (2005a) and McDonald et al. (2005b) introduce a highly effective model for scoring non-projective dependency trees, which we use as the g_d component of our factored model. Each tree is assigned a score that is the sum of the independent scores of the individual dependency edges in the tree. For the sake of uniformity, root tokens will be scored as if they were dependent on a special hidden 'ROOT' node: this means that in every tree there are effectively as many dependency edges as there are tokens.

The score of an individual edge e is given by the formula:

$$score_e = \mathbf{f_e} \cdot \mathbf{w}$$

where $\mathbf{f_e}$ is a vector of binary *feature values* (each value is 1 if the dependency has the feature, 0 if not) and $\mathbf{w}$ is a vector of *feature weights*. There are millions of features in McDonald et al.'s model, but only a few of them are expressed for any given edge. The features will be described below, under the heading Feature Set.

The total score of a dependency tree T is given by:

$$g_d(T) = \mathbf{f_T} \cdot \mathbf{w}$$

where $\mathbf{f_T} = \sum_{e \in T} \mathbf{f_e}$

Scoring Heuristic

It is quite feasible to devise a prediction heuristic for the value of g_d, to be used by our A* chart parser. Recall that our dependency scoring model assumes exactly one dependency edge per token position: each token will be the destination of one edge. Each chart arc includes a partial dependency function, so the edges (and their scores) associated with some positions may already be known exactly. The score of the edge associated with each position x among the remaining positions can be bounded above by choosing the maximum score over all possible edges from any position in the sentence to x.[1]

Feature Set

The base set of feature patterns used is given in table 3.1. Every combination of values *seen in the training data* that matches the fields of a feature pattern is considered one feature. So, for example $(p\_w = haben, c\_p = PPER, dist = -1)$ could be a feature matching the pattern $(p\_w, c\_p, dist)$, as could be $(p\_w = Mai, c\_p = APPRAT, dist = -2)$.

The features used in our model differ from those used by McDonald et al. in that we use only a subset of their features: in particular McDonald et al. use not only word-based fields (p_w, c_w) in the construction of patterns, but also 5-character-prefix-based fields.

Training

To estimate the values in the weight vector for dependency scoring, we could use the single-best MIRA training technique used by McDonald et al. (2005b). MIRA was introduced for online large-margin learning over a general set of problems by Crammer and Singer (2003); Crammer et al. (2004).

Briefly, the method works as follows. The weight vector is initialized to the zero vector. A sentence from the training data is parsed using the non-projective maximum spanning tree (MST) method of McDonald et al. (2005b). If the resulting tree (T) does not match the gold standard tree (T^*) for the sentence, the weight vector is adjusted so that the correct tree T^* would have outscored T by a value at least as large as the number of wrong dependency edges in T (other margins are possible, this one is referred to as the *hinge loss*.) Of all possible adjustments establishing the desired margin, the adjustment

[1]Tighter heuristics are possible, for instance taking the maximum spanning tree over unattached tokens, but I will use this simple one for my experiments.

base features
`dist, p-w, p-p, c-w, c-p`
`dist, p-w, p-p, c-w`
`dist, p-w, p-p, c-p`
`dist, p-w, p-p,`
`dist, p-w`
`dist, p-p`
`dist, p-p, c-w, c-p`
`dist, p-w, c-w, c-p`
`dist, c-w, c-p`
`dist, c-w`
`dist, c-p`
`dist, p-w, c-w`
`dist, p-w, c-p`
`dist, p-p, c-w`
`dist, p-p, c-p`
between features
`dir, p-p, b-p, c-p`
context features
`dir, p-p, p-p+1, c-p, c-p+1`
`dir, p-p-1, p-p, c-p, c-p+1`
`dir, p-p, p-p+1, c-p-1, c-p`
`dir, p-p-1, p-p, c-p-1, c-p`

Table 3.1: Dependency edge scoring features:
`c-w` is the child (dependent) word type, `c-p` is child part of speech, `p-w, p-p` are parent (governor) word and part of speech, `dist` is the position of the child minus the position of the parent, `dir` is the sign of `dist`, `b-p` is the part of speech of a token between the parent and child, `p-p+1, p-p-1, c-p+1, c-p-1` are the parts-of-speech of the tokens (immediately) after the parent, before the parent, after the child and before the child.

with the smallest Euclidean norm is chosen; this is fundamental to maintaining the theoretical properties of MIRA. In other words, given an original weight vector $\mathbf{w}$, we find an updated vector $\mathbf{w}'$ that is the solution to the optimization problem:

$$\text{minimize:} \quad |\mathbf{w}' - \mathbf{w}|$$

$$\text{subject to:} \quad \mathbf{w}' \cdot \mathbf{f}_{T^*} - \mathbf{w}' \cdot \mathbf{f}_T \geq \mathit{NumDifferentEdges}(T, T^*)$$

After the weight vector is adjusted, the next training sentence is parsed and the process is repeated. Training continues over the full set of samples in multiple passes until parsing accuracy on a development test set stabilizes.

The final weight vector output by this process may fail to provide good results on new data, because updates made on later examples may have led to a model no longer capable of correctly parsing earlier examples. Collins (2002) demonstrates that using the average of the weight vector over the full course of training can improve the generality of a discriminative model trained incrementally. This technique was carried over in the work of McDonald et al. (2005a) and McDonald et al. (2005b), and I also use it.

3.4.3 Phrase Structure Model

PCFG with Treebank Transformation

The phrase structure scoring function (g_p) is based on a probabilistic context-free grammar (PCFG) model (a rapid introduction to PCFG can be found in textbooks such as Manning and Schütze, 1999). Each production rule is assigned a probability of being chosen to expand its left-hand-side symbol, and the score of each tree T is the base-10 log of the probability of T being generated stochastically by the grammar from the start symbol S. The main difference between our model and traditional PCFG lies in what constitutes a production rule. In context-free filtering grammar each rule has the form:

$$L \leftarrow R_0^{e_0} R_1^{e_1} ... R_n^{e_n}$$

where $L \in \mathcal{N}$, $R_i \in \mathcal{N} \cup \mathcal{T}$ and $e_i \in \{Hidden, Exposed, Unlinked\}, \forall i$ from $0..n$. The exposures are an integral part of the rule, and otherwise identical rules with differing exposure labellings may be generated with different probabilities.

Scoring Heuristic

Since g_p is essentially a PCFG score, we can use the same heuristic for the phrase structure component as Klein and Manning (2002): the (log) joint probability of the production rules used in the derivation of the chart arc plus the outside score of the arc's constituent. All possible outside scores can be precomputed using the CKY algorithm before starting the chart parser.

Rule Scoring Details

To improve the scoring function's granularity, parent encoding is used. Parent encoding is the practice of labeling every non-terminal constituent not only with its category, but also with the category of its parent. Thus the non-terminal set of the PCFG is the set of all *child_cat*^*parent_cat* pairs seen in the training data. The substantial benefits of parent encoding for pure PCFG models are pointed out by Johnson (1998), and elaborated upon by Klein and Manning (2003). The number of ancestors to consider is a parameter that can be tuned against a development corpus.

The Klein and Manning (2003) study describes parent annotation as taking advantage of vertical history to make constituent generation decisions, to distinguish it from taking advantage of the horizontal history of decisions. The horizontal history refers to the siblings generated between a current decision point and the head child of a constituent. Both Johnson (1998) and Klein and Manning (2003) point out that PCFGs stripped naïvely from corpora encode too great a dependence on horizontal history (i.e. probabilities are defined over complete right-hand-sides of production rules), resulting in sparse data problems.

This issue can be alleviated by *Markovizing* the model, so that children are generated individually (conditioned on their head and nearby inward siblings), rather than en-masse.[2] The Markovization of productions is also used in this study. The number of siblings to consider in a Markovized model is a parameter tunable against a development corpus. To illustrate the model used in this study, I will describe it in the particular case where up to one sibling is considered. Given a parent constituent of category L:

1. the model generates a head child of category H with probability $Pr(H|L)$;
2. the exposure of the head is pre-determined ($Unlinked$ if $L = S$, $Exposed$ otherwise);
3. on the left side of the head, no siblings may be generated, with probability $Pr(\text{STOP\_LEFT}|H, L)$;
4. if an inner-most sibling is generated, its category C_1 and exposure e_1 are chosen with probability $Pr(C_1, e_1|H, L)$;
5. before generating each additional ith sibling, the process may stop with probability $Pr(\text{STOP\_LEFT}|C_{i-1}, H, L)$;
6. each ith additional sibling with category C_i and exposure e_i is generated with probability $Pr(C_i, e_i|C_{i-1}, H, L)$;
7. symbols to the right of the head are generated in the same manner as those to the left.

Although context-free filtering grammar permits multiple exposed right-hand-side children, the Markovized PCFG model assumes one distinguished head. In any rule with

[2]Note that Markovization of rules can be implemented as a grammar transformation. This was in fact done in my parser implementation. The new grammar has only binary rules and has more general coverage than the original, but remains a PCFG. If Markovization by grammar transformation is used, then parse-trees generated according to the new grammar need to be detransformed back into the original format to eliminate the 'rule-internal' non-terminals that were not present in the original grammar.

exactly one exposed child, that child is the head, otherwise we take the leftmost exposed child to be the head (or for top-level rules, the leftmost unlinked child.)

Grammar Extraction

The production rules can be stripped from a parallel corpus of phrase structure and dependency trees, in essentially the same way as with normal context-free grammars. The only complicating factor is that each right-hand-side symbol of a rule needs to be annotated with an exposure, and treebanks do not come annotated with constituent exposures. We therefore need to be able to perform this annotation ourselves, automatically, before extracting the grammar.

This annotation decision is simple to make for each constituent. Consider all tokens in the yield of a constituent c:

- if no token is dependent on a token outside c, then constituent must be *Unlinked*;
- if any token governs a dependent outside c, then c must be *Exposed*;
- if any token is the dependent of another token not in the immediate parent of c, then c must be *Exposed*;
- otherwise c is *Hidden*.

In some cases, it may be preferable to assign exposures based on linguistic knowledge about the category of c. In the experiment done as part of this research, the TüBa-D/Z corpus was used, which includes topological field constituents. I choose to mark all children of each topological field as *Exposed*, and rely on the more complicated heuristic just described for all other categories of constituent.

As with regular PCFGs, we count the number of occurrences of each rule (keeping in mind that rules may be distinguished by differing exposures, even if the right-hand-side symbols are otherwise identical), and use the counts to compute estimates for all Markovized rule generation probabilities.

3.4.4 Factored Model Training

The base training method for the dependency-scoring weight vector does not take advantage of phrase structure information: the weight vector is trained as if the model had to be able to choose from among all possible dependency trees. In our factored model, however, the weight vector will only really have to choose among dependency trees licensed by high-scoring phrase structure trees.

The approach taken in this work in order to exploit the reduced responsibility of the dependency model, is to modify both the selection of the training tree T and the update procedure used in training. In the base training process, T is chosen using an MST dependency parser. In the factored trainer, T is chosen using A* search over the full factored scoring model. The update procedure will work differently for two cases:

1. if the phrase structure (call it P) returned with T licenses the correct dependency tree T^*, or is outscored by a phrase structure P^* that does, then the original update procedure is used;

2. otherwise we find the highest scoring phrase structure (call it P_h) that does license the tree T^* and we update by solving the optimization problem:

$$\text{minimize:} \quad |\mathbf{w}' - \mathbf{w}|$$

$$\text{subject to:} \quad \mathbf{w}' \cdot \mathbf{f}_{T^*} - \mathbf{w}' \cdot \mathbf{f}_T \geq 1.0 + g_p(P) - g_p(P_h)$$

In the first case, the phrase structure found by the factored parse is correct — g_p has contributed as much as it can. In the second case, g_p predicted the wrong phrase structure, and g_d must be modified to compensate for the error. The constant factor of 1.0 ensures that the new classifier will give a better score to the correct tree than the found tree, rather than simply an equal score.

3.5 Summary

In this section I have presented context-free filtering grammar, a generalization of the grammar required by the factored model of Klein and Manning (2002). The generalization works by permitting more than a single token to be 'exposed' by each constituent. I demonstrated some of the representational advantages of the new model using an example from the German TüBa-D/Z corpus. I extended the generalization to handle potentially unconnected dependency structures and I outlined a chart-parsing method to recover parse trees according to a context-free filtering grammar.

Having provided a new mechanism for encoding the constraints between phrase structure and dependency structure, I proposed a statistical model and parsing method derived from Klein and Manning (2002). The new model combines the dependency model of McDonald et al. (2005b) with a PCFG model similar to that of Klein and Manning (2003). I provided an admissible and monotonic heuristic for the new factored model, suggesting that the A* search that works so well for Klein and Manning (2002) will also work for the new model. Finally, I introduced a method for training the new model to take advantage of the interaction between phrase structure and dependencies.

Chapter 4

Experimental Design

4.1 Overview

In this chapter, I describe my approach to testing whether or not the potential strength of a *factored-model statistical parser built on context-free filtering grammar* (henceforth the *experimental model*) actually manifests itself when parsing real-world data; in particular sentences of the German TüBa-D/Z treebank.

Context-free filtering grammar can be argued to be of potential interest in future research projects on one of two grounds:

1. the new formalism permits a parser to reliably generate more useful syntactic analyses than was previously possible; or

2. the new formalism improves parsing reliability on tasks of which already-existing systems are capable.

The experiment outlined here aims to establish 2. We should not forget 1, however. Context-free filtering grammar does allow a simultaneous and complementary phrase structure and dependency analysis that is not possible in previous statistical parsing systems. As the experimental results of the next chapter demonstrate, the new system provides this enhanced output without compromising accuracy compared to either pure phrase structure or pure dependency parsers.

We would expect the experimental model to 'outperform' a pure phrase based parser or a pure dependency parser. This expectation is based on the original finding of Klein and Manning (2002) that their factored model outperforms its component phrase and dependency models on the Penn treebank, and on the finding by Kübler et al. (2006) that the same factored model outperforms its phrase structure component on the TüBa-D/Z treebank. This latter result comes despite the fact that the factored model used by Kübler et al. is not modified to handle topological field constituents differently from phrase constituents: each field is treated as a headed phrase in which the left-most child is the head (personal communication with the authors.) Since these 'heads' are not systematically linguistically justifiable, their relations to their dependents and governors are possibly quite arbitrary and unpredictable. The experimental model does not restrict field constituents to having only one token in dependency relations with other parts of the sentence, and can model more realistic dependency relations as a result. We can hope that the improved dependency modeling will lead to a further improvement in accuracy by the experimental model over the original factored model.

The essence of the experiment is as follows:

1. I do some preliminary tests to tune a number of parameters, on a development set drawn from the German TüBa-D/Z treebank;

2. I run tuned versions of a pure phrase structure parser, a pure dependency parser, a factored parser in the style of Klein and Manning (2002), and a factored parser with the experimental model, on a test set drawn from the German TüBa-D/Z treebank;

3. I determine whether the experimental model outperforms the other three.

4.2 Hypotheses

The parsing system outlined in the previous chapter aims to combine the strengths of context-free parsers and dependency parsers. Accordingly, we would expect the following.

1. The experimental model will produce better phrase structure trees than a well-tuned unlexicalized PCFG parser.

2. The experimental model will produce better dependency trees than a well-tuned dependency parser.

3. The experimental model will produce better phrase structure trees than a well-tuned (but context-free) lexicalized parser.

In each case, the null hypothesis is that the experimental model will either underperform the other parser, or that any improvement shown by the experimental model can likely be attributed to chance.[1] Although demonstrating that the null hypothesis can likely be rejected is arguably due diligence in any experiment, in this case we are more interested in being able to conclude that the new system is better than existing systems to a degree large enough to justify its additional complexity.

In the remainder of this chapter, I flesh out what I mean by 'better' and I address more clearly how the parsers are tuned and how their performance is measured. Unfortunately, comparisons between parsers will have to rely on simplistic quantitative measures of their ability to reproduce gold-standard data. Comparisons based on a deeper analysis of results are difficult to justify except in relation the specific needs of an application, and this work was not carried out in the context of any particular application (industrial or academic.)

[1]What 'chance' actually means in this context is an interesting question, and one that will be partially addressed in section 4.3.4.

4.3 Challenges

The experiments of this chapter were designed under a number of constraints that potentially reduce our ability to spot meaningful differences between the parsing strategies. Many of these constraints are imposed by mismatches in the representational assumptions of the TüBa-D/Z corpus annotation and of the parsers. In principle, the parsers could be augmented to the point where all of the information available from the corpus can be used, but the development effort might not be warranted if we can detect obvious failures or successes of the models without these enhancements.

4.3.1 Oversimplifications in the Syntactic Representation

A source of concern in this study is that syntactic representations used by all of the parsers admit some ambiguities. The phrase structure parsers produce context-free output matching the node-labels of the TüBa-D/Z corpus, but do not output the edge-labels of the corpus. These edge labels encode grammatical functions, especially verb-argument structure. The labels also encode dependencies between phrases in different topological fields. In their absence we are unable to directly evaluate the parsers' sensitivity to these linguistic features.

The dependency trees used in this experiment are unlabeled, so again there is significant ambiguity in the grammatical relations represented in a single tree. Furthermore, co-ordination is notoriously problematic to represent in dependency trees, and the lack of labels only worsens this problem.

We might expect, however, that models that better capture information about grammatical function will be able to outperform models that do not, even according to parsing metrics that do not directly measure a parser's ability to report these relationships. A classic example of this is the use of lexicalization in PCFG-parsing: a parsing model leveraging lexical information (an indirect source of knowledge about grammatical function) can score better on constituent precision and recall than a model that does not, even though constituent precision and recall do not measure a model's ability to recover lexical dependencies.

4.3.2 Lack of Gold-Standard Dependency Trees

There are no published dependency trees for the TüBa-D/Z corpus (although Kübler and Prokić, 2006, recently generated their own). We are therefore constrained to using trees produced from the phrase structure treebank using heuristic rules. Since these automatically generated dependency trees have not been assembled (or even verified!) by trained linguists, their usefulness is at best hypothetical. We can, however, retroactively impute some degree of validity to these trees if a parser having access to them can outperform a parser that does not, when evaluated with respect to the gold standard phrase structure trees.

4.3.3 Measuring Performance

What it means for one parser to outperform another is a troubling question. Ideally, we would want to be able to compare the suitability of different parsers as components in a variety of user applications. But it is often the case, as it is here, that statistical parser development is undertaken independently of any larger application. Under these conditions, it is necessary to speculate as to what behavioural properties of a parser could be of interest to an application developer and to measure these properties. Traditionally, the Parseval metrics of Black et al. (1991) have been used to evaluate phrase structure parsers, and governor identification accuracy has been used to evaluate unlabeled dependency parsers (if a governor is assigned to every terminal except the one root, then precision and recall are the same.)

I report phrase structure parsing accuracies according to the following metrics:

coverage: the proportion of sentences for which the parser produced a parse tree.

perfect: the proportion of gold-standard sentences for which the parser produced exactly the correct tree.

mean crossing brackets: the number of crossing brackets (defined according to Black et al., 1991) divided by the number of sentences parsed.

0 crossing brackets: the proportion of sentences parsed that contain zero crossing brackets.

micro-averaged labeled constituent precision: the total number of constituents *correctly* found by the parser (over the entire test corpus) divided by the total number of constituents output by the parser. The special root constituent and the pre-terminal constituents are not included in the counts.

micro-averaged labeled constituent recall: the total number of constituents *correctly* found by the parser (over the entire test corpus) divided by the total number of constituents in the gold-standard results. The special root constituent and the pre-terminal constituents are not included in the counts.

Of these metrics, I pay special attention to the perfect score: although it is a harsh metric, there can be no question of what it means. An exact match with a gold-standard parse is as useful to an application as the gold-standard parse would have been. Partial sentence matches, that receive points under other metrics, may be more or less useful, depending on the needs of the application and on the precise nature of the error made. The perfect score also shows a large amount of variability between different parser configurations, and relying on it for tuning reduces the odds of having to resolve a tie.

One problem that the perfect score does have is that it will tend to emphasize performance on shorter sentences (which are easier to parse correctly) and will not reflect the reasonableness or unreasonableness of behaviour on longer sentences.

I report dependency parsing accuracies according to these metrics:

perfect: the proportion of gold-standard sentences for which the parser produced exactly the correct tree.

micro-averaged non-root attachment precision: the total number of *correct* dependency edges produced by the parser (over the entire test corpus, and excluding dependencies incident to the special Root token) divided by the total number of dependency edges produced by the parser (excluding those incident to the Root token.)

micro-averaged non-root attachment recall: the total number of *correct* dependency edges produced by the parser (over the entire test corpus, and excluding dependencies incident to the special Root token) divided by the total number of dependency edges in the 'gold-standard' results (excluding those incident to the Root token.)

Again, I pay special attention to the perfect score. I elect to ignore edges incident to the Root token, because a large number of the tokens dependent on the Root are punctuation symbols, and attaching them correctly is trivial. An alternative approach would be to ignore punctuation explicitly. Ideally the punctuation would not be in the trees at all, but it is included in the phrase structure trees and treating every punctuation mark as 'just another token' simplifies the task of gathering model features sensitive to punctuation.

As a final note, rather than comparing *parsers*, we might consider comparing *models*. Two probabilistic models can be compared according to the likelihood that they assign to previously unseen correct parse trees. The dependency component of my parser, however, is not a probabilistic model. It would still be possible to compare the average rank assigned to each unseen correct parse tree (in relation to other parse trees considered by the model for the same input sentence), were it not for the pruning performed by my search algorithm.

4.3.4 Establishing Statistical Significance

In comparing the metrics of results from two parsers, A and B, where we wish to claim that the results of B are better than those of A, and having seen sample results in which B does outperform A, we wish to be able to determine how likely it is that B's results were better than A's results simply 'by chance'.

But what do we mean by 'chance'? In the statistical parsing community, parsers are developed, trained and evaluated against an annotated corpus. The corpus is seen to have been created by an unknown random process R. Before development and training of the parsers, a test set is put aside from this corpus, to be ignored until the time comes to evaluate the parser. If this sample is not examined until after the parsers have been developed and trained, then we claim to be able to treat the unseen sentences as having been newly generated by R.

Suppose we expect parser B to produce better results than parser A. We attempt a proof by contradiction: we assume that the results of A are as good as the results of B over the distribution of the random process R. We ask what the likelihood is of seeing the results we achieved, over a random sample generated by R. If the answer is "not very likely at all," we conclude that the assumption must have been false, and that B really is better than A. This conclusion comes with the caveat that it isn't strictly impossible for us to have seen the results that we did, and we report an upper bound on the probability that we could have been wrong (the significance.)

Establishing significance values over some of the metrics outlined above in section 4.3.3 is difficult to do analytically. Instead of trying to find a well understood distribution for these values, and approximating significance based on this distribution, I have opted to use a paired permutation test (designed according to Good, 2000, p.25). In brief, the permutation test works by estimating the distribution of results over R under the assumed hypothesis that A is as good as B. If we assume:

1. that the expected value of per-sentence evaluation scores of A parsing sentences from R is the same as the expected value of the per-sentence evaluation scores of B parsing sentences from R (the null hypothesis);

2. that the results on different sentences are independent; and

3. that the results of A parsing sentences from R have the same distribution (differing in mean, possibly, but not in shape or variability) as of B parsing sentences from R;

then we can estimate the distribution of the difference in the statistics between A and B over different samples from R by looking at all possible rearrangements of the results over the sample we have.

Consider the results we have for a single sentence. If the results for that sentence generated by A and generated by B are random and drawn from the same distribution (which we are assuming), then they differ only by a random error term. The error generated by A could just as easily have been generated by B, and visa versa. We can ask what the overall statistics for A and B would have been if the errors had been drawn in reverse. If we ask this question for every subset of sentences in the test set, then we wind up 2^n new samples differing only in their assignments of random measurement errors. Computing the difference in statistics over all of these new samples gives an estimate of the behaviour of the difference in statistics over any sample from R, under the assumptions above.[2] We can estimate the significance of the original finding as the proportion of differences, over the permuted results, in which B beats A by more than it did in the original results.

[2]In practice, I will sample permutations at random 10,000 times, rather than trying all possible permutations.

The first assumption above is the null hypothesis that we hope to contradict, with an upper bound on the significance of the conclusion. But we can only rely on the significance of the conclusion if we can be certain that the other two assumptions hold.

In most statistical parsing research, the test set consists of a contiguous block of text, and it is widely believed that measurements of syntactic parsing performance on sentences from such a block will not be entirely independent. I am not aware of any quantification of how bad this problem is: how large a sample would be required for an uninformed observer to be able to detect the dependence? Given that I drew my sample at random from the corpus, any dependence will be even further diluted, but the fact remains that the independence assumption cannot be justified.

We also have no reason to believe that the third assumption holds. Perhaps parser A consistently achieves some degree of correctness, while B either gets a sentence all right or all wrong. Perhaps one parser is more sensitive to the length of a sentence than the other. This problem is potentially quite damaging. In the case of simple means (e.g. the perfect score), we can wrest some solace from the result cited in Good (2000) on pg. 212 (Theorem 7): that as the sample size runs to infinity, the significance value produced by the permutation test will asymptotically approach the exact value, even if the distributions from the two parsers are not of the same shape (independence of measurements must still hold, however.)

In sum, the 'significance' values that I will be computing can only be treated as a rough approximation of the probability that we can reject a null hypothesis. A skeptic (or a statistician) would probably go so far as to call these values entirely worthless and misleading, but I am aware of no better process for detecting whether or not the results have any potential validity.

4.4 Corpus Data

The treebank corpus used in any experiment involving statistical parsers is of critical importance: the treebank is needed both to train the parsers and to evaluate their output. In the following paragraphs I will review briefly the properties of the corpus used in this experiment, the TüBa-D/Z corpus, and a number of additional resources we can derive from this data.

4.4.1 TüBa-D/Z Description

The TüBa-D/Z treebank corpus contains just over 22,000 sentences of German newspaper (‘die Tageszeitung’) text, for a total of over 380,000 tokens, including punctuation marks. Each sentence is annotated with context-free phrase structure. Constituent annotations within noun phrases and prepositional phrases will appear familiar to users of earlier Penn treebank-style corpora, with the addition of *edge-labels* to mark which constituents are head constituents. It is immediately below the clause-level where significant differences are to be found: rather than being divided into a subject and verb-phrase, TüBa-D/Z clauses decompose into topological fields which contain the various verbal constructs, arguments and adjuncts. In a main declarative clause, one can expect to find the following fields (in order):

Vorfeld: the first field contains one phrase, often the subject;

linke Satzklammer: the left sentence bracket contains the finite verb of the clause or an auxiliary;

Mittelfeld: the middle field can contain any number of phrases, usually arguments and adjuncts of the verb;

Verbkomplex: the right sentence bracket, also known as the verb complex, contains a finite verb, one or more participles or a verb particle;

Nachfeld: the final field, if present, often contains longer phrases, especially relative or subordinate clauses (some clause types can *only* occur in the Nachfeld.)

Since verb-argument structure is not explicit in the nesting of these phrases, grammatical functions are indicated by means of the *edge-labels* on all constituents immediately below the field level.

The TüBa-D/Z annotation is unusual in the number and kinds of sources of constituent labels. In many corpora, terminal nodes are labeled with part of speech tags, while internal nodes are labeled with phrase types (e.g. noun phrase, prepositional phrase.) This is also true with TüBa-D/Z , but in addition to internal nodes bearing phrase labels, there are also nodes bearing topological field names. It is worth keeping in mind that although the parsers will be evaluated on their ability to retrieve 'correctly labeled constituents', the labels do not represent one property of syntax but rather three: part of speech, semantically coherent phrase type, and topological field. Performance on these three kinds of properties could be evaluated independently (indeed, part of speech *is* handled separately: it is excluded from consideration.)

An example parse tree from TüBa-D/Z is shown in figure 4.1. For a fuller account of the treebank annotation, please refer to Telljohann et al. (2005).

The treebank has one peculiarity which must be addressed: parse trees are not always connected (the tree in figure 4.1 is disconnected, for example.) Each punctuation mark is represented by a free-floating token with no parent constituent. A parenthetical element may likewise be left unattached, with no parent. Multiple independent clauses, if not explicitly co-ordinated, may be represented as disconnected constituents. I eliminate this disconnection by adding a top-level root constituent to every sentence, and by projecting all unattached constituents (including punctuation) upward as far as possible until each one is embedded in some higher-level parent (example in 4.2.)

To the reader unfamiliar with TüBa-D/Z , it might be helpful to take in a few of the quantitative properties of the corpus (the following were all computed over the ex-

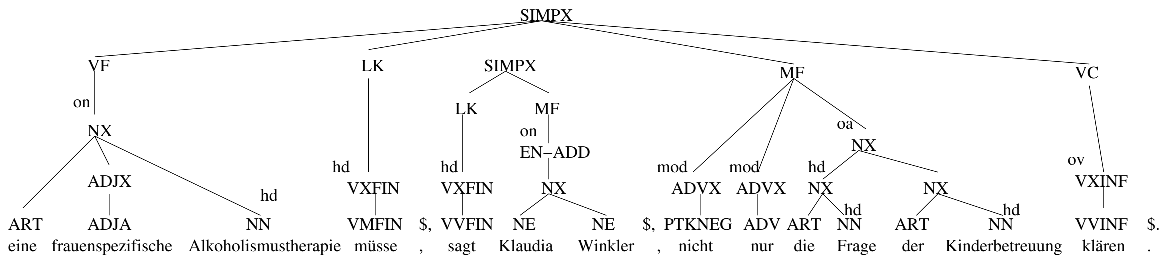
SIMPX
VF
on
NX
ADJX
hd
LK
hd
VXFIN
SIMPX
LK
hd
VXFIN
MF
on
EN-ADD
NX
MF
mod
ADVX
mod
ADVX
oa
NX
hd
NX
hd
NX
hd
VC
ov
VXINF
ART ADJA NN VMFIN $, VVFIN NE NE $, PTKNEG ADV ART NN ART NN VVINF $.
eine frauenspezifische Alkoholismustherapie müsse , sagt Klaudia Winkler , nicht nur die Frage der Kinderbetreuung klären .

Figure 4.1: A disconnected phrase structure tree from TüBa-D/Z .

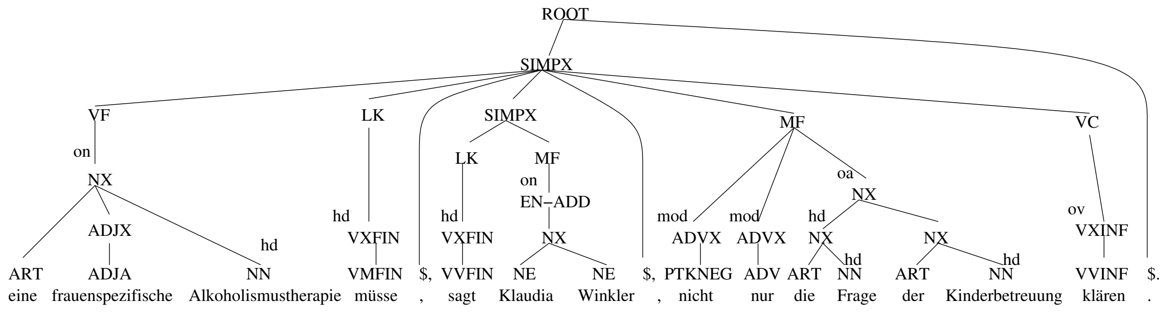
ROOT
SIMPX
VF
on
NX
ADJX
hd
LK
hd
VXFIN
SIMPX
LK
hd
VXFIN
MF
on
EN-ADD
NX
MF
mod
ADVX
mod
ADVX
oa
NX
hd
NX
hd
NX
hd
VC
ov
VXINF
ART ADJA NN VMFIN $, VVFIN NE NE $, PTKNEG ADV ART NN ART NN VVINF $.
eine frauenspezifische Alkoholismustherapie müsse , sagt Klaudia Winkler , nicht nur die Frage der Kinderbetreuung klären .

Figure 4.2: A connected phrase structure tree derived from TüBa-D/Z .

periment's training set.) The average sentence is 17.2 tokens long, 14.7 tokens long if punctuation is omitted. 87.0% of sentences have at least one clause.[3] 42.1% of sentences have more than one clause. 18.8% of sentences contain a clause without a subject and 10.9% of sentences, a clause without its own finite verb.[4] 31.5% of sentences contain some form of co-ordination, and in 8.2% of sentences entire fields or groups of fields are co-ordinated.[5] In the dependency trees derived from the corpus (see section 4.4.3, below) 12.5% of sentences contain at least one non-projective dependency.

[3]what I have been calling a *sentence* is more accurately called an *utterance*, since it includes short non-sentential phrases appearing in the newspaper text.

[4]note, however, that a clause formed by the co-ordination of two other clauses will be included in this count.

[5]co-ordinated fields occur when two 'clauses' share field elements, especially the Vorfeld.

4.4.2 Corpus Split

Before development of the parsers begins, the TüBa-D/Z treebank is split into three subsets: a training set comprising roughly 90% of sentences (19,856 of them), a development set comprising roughly 5% of sentences (1,109 of them) and an test set comprising roughly 5% of sentences (1,126 of them.) Because of computing resource (memory) limitations, only sentences with 40 or fewer tokens, including punctuation, are used in evaluation (1,070 in the development set and 1,086 in the test set.) The same limitations apply to the factored model training, which is only performed on sentences of 40 or fewer tokens.[6] The grammars extracted for this experiment are extracted from the full set of training sentences and the pure dependency models are also trained over the full set of training sentences.

The dependency tree extraction scripts, grammatical formalism and parser implementations are all developed in consultation with the training set. The statistical models are trained on the training set; the development set is used for debugging the parser implementations. Further uses of the development and test sets are described in the remainder of this chapter.

The split is done randomly, each sentence being assigned to one of the three subsets independently of all other sentences. If we were intending to evaluate the performance of a parser in absolute terms, this split potentially gives an artificial boost to scores by including in the test set sentences that come from the same documents and paragraphs on which the parser was trained and tuned. Of course, this is but an exacerbation of the fact that we are training and evaluating on text derived from the same underlying publication ('die Tageszeitung'.) I should not expect this to negatively reflect on an effort to compare the parsers within this study however: they all have the same potential advantage.

[6]The table of precomputed outside scores used in the phrase-structure scoring heuristic is the main user of memory and the primary reason for a limit on input length.

Moreover, drawing test set examples at random from the entire TüBa-D/Z corpus increases our chances of achieving representative coverage of the full range of linguistic phenomena in the treebank. The selection method may also have implications for computing the significance of differences between parsers (see section 4.3.4.)

4.4.3 Dependency Tree Extraction

The dependency treebank used in this experiment is extracted from the TüBa-D/Z phrase structure corpus by a rule-based script. As much as possible, I try to rely directly on information provided by the phrase structure and by the edge-labels from the original data.

In broad terms, the script works bottom-up. Each terminal node takes its lexical item as its head. Each non-terminal node, once its children have been assigned heads, finds the child with edge-label 'HD' and takes that child's lexical head as its own lexical head; all other children have their lexical heads assigned as dependents to this parent head.

This general strategy is not applicable in a number of circumstances (and thankfully too, since this is what makes the dependency trees valuable.) A clausal constituent ('SIMPX', 'R-SIMPX' or 'P-SIMPX') requires the most interesting treatment. Recall that the immediate children of a clause are field constituents. The script first processes those fields containing verbal elements (the linke Klammer 'LK' and the Verbkomplex 'VC'): the finite verb (or auxiliary) is selected as the *verb-head*, and any remaining verbs are chained together according to the annotation available in the Verbkomplex. The final dependent in this chain of verbs is the main verb of the clause. The *verb-head* and main verb could be the same token, if there is only one verbal token in the clause. The script next looks at the phrases within the remaining fields in the clause. Each of these phrases has one lexical head and an edge-label that marks its grammatical function. The script uses this grammatical function to find the governor of the head according to table 4.1. The *verb-head* is then taken to be the head of the clause.

child label	**potential governor labels**
ES	*verb-head*
ON (nominative-object)	*verb-head*
OD (dative-object)	*main verb*
OA (accusative-object)	*main verb*
OG (genitive-object)	*main verb*
OS (sentential-object)	*main verb*
OPP (prepositional-object)	*main verb*
OADJP (adverbial-object)	*main verb*
OADVP (adjectival-object)	*main verb*
PRED (predicate)	*main verb*
FOPP (passivized subject)	*main verb*
MOD (modifier)	*verb-head*
ON-MOD	ON
OA-MOD	OA
OD-MOD	OD
OG-MOD	OG
OPP-MOD	OPP
OS-MOD	OS
PRED-MOD	PRED
FOPP-MOD	FOPP
OADJP-MO	OADJP
OADVP-MO	OADVP
V-MOD	*main verb*
MOD-MOD	MOD,X-MOD

Table 4.1: Governor selection for the immediate children of a clausal constituent. In some cases the labels may not be sufficient to disambiguate dependency attachments. When this happens the TüBa-D/Z corpus supplies the attachment information explicitly.

The dependency representation I use for co-ordination is based on Mel'čuk (1988): the head of the first conjunct is the head of the co-ordinated phrase, the conjunction itself is a dependent of this first head, and all conjuncts after the first are dependents of the conjunction. In the case of co-ordinated fields, the phrases within the first conjunct field are available to participate in relations with other elements of the enclosing clause, but phrases in subsequent conjuncts are not. The conjunction is a dependent of the most dominant verb in the first conjunct (usually a finite verb) if the conjunct contains any verbs, otherwise the conjunction is a dependent of the head of the first phrase of the first conjunct. Subsequent conjuncts are processed as if they were clauses, and then linked to the conjunction. Treating later conjuncts as clauses leads to relatively intuitive structures, unless these conjuncts contain no verb, in which case the results can be arbitrarily bad (this, thankfully, is a relatively rare occurrence.)

As mentioned above, the original TüBa-D/Z trees may be unconnected. As with the phrase structure trees I use for this experiment, I eliminate the disconnectedness in the dependency trees by adding a special Root token. The heads of all free-floating constituents are treated as dependents of this special token (this includes, for instance, all punctuation symbols.)

Figure 4.3 shows an example of the final result of the dependency extraction process.

4.4.4 Phrase-Head Tree Extraction

In addition to the proper dependency trees used in the experiment, we also need trees of linked phrase heads. These phrase-head trees are required to train the Klein-and-Manning-styled (2002) factored lexicalized parser, against which the experimental model is to be compared.

The phrase-head trees are formed by assigning to each constituent a head token. The head token must be the head of one of the constituent's children, and the heads of all other children are linked directly to this one head token.

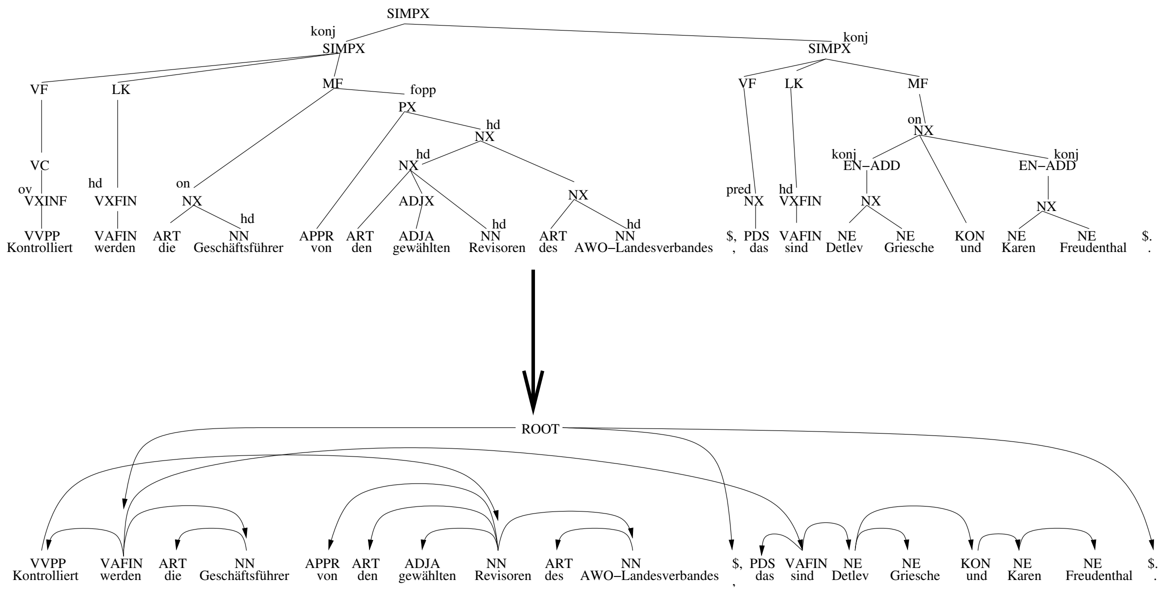

Figure 4.3: Extraction of a dependency tree from a TüBa-D/Z tree.

Each terminal node takes its lexical item as its head. Phrasal non-terminals have a head child marked in the original corpus. Fields do not have heads marked, so the head of the first child of each field is taken as that field's head. The head of a clause is one of: the finite verb, the first verb in the Verbkomplex, or the head of the first child of the clause (in descending order of priority.) In the case of co-ordination, the head of the first conjunct is taken as the head of the co-ordinated phrase.

Free-floating constituents are handled in the same manner as with the dependency trees: the heads of all free-floating constituents are linked as dependents to a special Root token.

Figure 4.4 shows an example of the final result of the lexical heads extraction process.

4.5 Parsing System

The parsing system used in this experiment is written specifically for this experiment, although in principle, given enough massaging of input formats, it could be used on any

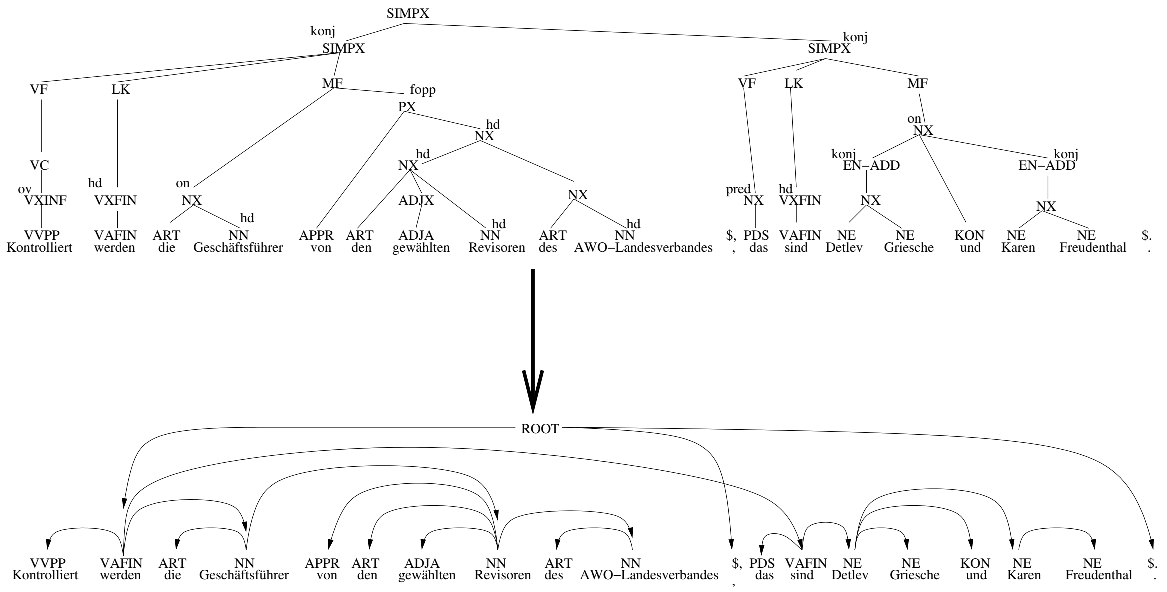

Figure 4.4: Extraction of a lexical heads tree from a TüBa-D/Z tree. Note in particular that the head of 'Revisoren' is 'Geschäftsführer', because 'Geschäftsführer' was chosen to be the head of the Mittelfeld, and that the co-ordination near the end of the sentence is handled differently than in figure 4.3.

paired phrase structure/dependency-structure treebank. In total, three parsing modes are supported: a pure phrase structure parser, a pure dependency parser, and a factored parser. The latter two modes can be used either to generate output for evaluation or to train a dependency model, as described in sections 3.4.2 and 3.4.4. The phrase structure mode only generates output for evaluation. In order to limit as much as possible the differences between the algorithms of the different parsing modes, the factored mode re-uses as many of the components of the two pure modes as possible.

All of the parsers require part of speech annotated input, and in this experiment I provide gold-standard parts of speech in the input. This will boost absolute scores above what would be possible if the parts of speech had to be determined automatically. If some parsing configurations are more robust to noisy part of speech input than others, then these differences may compromise my results.

The search algorithm that I use for the factored system is not a pure A* algorithm. The basic search is the same, but the system prunes arcs to avoid running out of memory on difficult sentences: I limit to 500 the number of arcs considered for each possible span and for each left-hand-side category.[7]Active and passive arcs are considered together, not separately. This pruning technique has two negative consequences: the algorithm is no longer guaranteed to find a solution if one exists, and even if a solution is found it is not guaranteed to be model-optimal. On sentences that are easily parsed, I expect the pruning to have little effect, but it will come into play when the dependency model and the phrase structure model cannot agree on an interpretation. It may be possible to greatly reduce the need for pruning by adopting a tighter scoring heuristic, in particular for the dependency portion of the model.

None of the parsing systems employ smoothing for parameter estimation.

[7]In runs against development data, subsequent to this experiment, I find that a limit of 500 arcs is unnecessarily aggressive. A limit of 800 arcs leads to significantly improved results, without incurring large increases in memory requirements.

	base grammar	**filtering grammar**	**no phrase structure**
dependency trees		experimental parser	dependency parser
lexical heads	lexicalized parser		lexical heads parser
no dependencies	PCFG parser		

Table 4.2: Parser configurations.

4.6 Grammars

Two context-free filtering grammars are used in the experiment. They are both built according to the grammar extraction algorithm proposed in section 3.4.3, but each with respect to a different dependency tree.

The first grammar is effectively a context-free grammar with one marked head per constituent: it is built using the lexical head trees as a dependency source. This grammar includes only one *exposed* child per rule, but may permit multiple *unlinked* children. If the grammar is used without a dependency model, I call the resulting parser a *(pure) PCFG parser*. If the grammar is used with a dependency model trained on the lexical head trees, I call the parser a *lexicalized parser*.

The second grammar takes full advantage of the flexibility of context-free filtering grammar: it is built using the proper dependency trees as a dependency source. This grammar permits multiple *exposed* children per rule, as well as multiple *unlinked* children. A parser run using this second grammar I call the *experimental parser*. This grammar is always used with a dependency model trained on the proper dependency trees.

I will sometimes refer to both the experimental and the lexicalized parsers as *mixed parsers*, since they use both a phrase-structure model and a dependency model.

A parser can also be run as a pure dependency parser without a phrase-structure grammar, in this case I call it either a *lexical heads parser* or a *(pure) dependency parser*, depending on which source of dependencies was used for training.

A summary of the possible configurations is given in table 4.2

4.7 Method

Finally, I am in a position to fully describe the experiment. The first stage of the experiment is to determine good values for the parameters left unspecified for the basic parsers in section 3.4. The second stage is to determine good values for the parameters used in the two factored parsers. The third stage, the only stage to be performed on the test set, is to generate results for all of the tuned parsers and to compare them.

For tuning purposes, I run a parser with a range of parameter values, and choose the parameter value for which the parser achieves the highest perfect score on the development data. I choose the perfect score for its simplicity and because during development it showed a relatively large variability across parsers.

All stages of the experiment use exclusively the TüBa-D/Z corpus, and the dependency trees heuristically extracted from it, as described in section 4.4.3. This corpus is first split into a training set, a development set and an test set, as described in section 4.4.2.

In the first stage, I choose the following parameter values.

- For the phrase structure parser:
  - the amount of vertical context to use in rule generation (values considered are none, parent encoding and grandparent encoding);
  - the amount of horizontal context to use in rule generation (values considered are 0, 1, 2, 3 and infinity.)[8]

- For the dependency parser:
  - the minimum number of times a feature must occur in training data before it is included in the model (values considered are 1, 5 and 10);

[8]Unlike Klein and Manning (2003), the model used here does not make left children dependent on right children, so a horizontal context size of infinity does not correspond exactly to an un-Markovized PCFG model.

  - the number of learning passes through the training data (values considered are 1 through 8.)

- For the lexical heads parser:
  - the minimum number of times a feature must occur in training data before it is included in the model (values considered are 1, 5 and 10);
  - the number of learning passes through the training data (values considered are 1 through 8.)

Although the lexical heads parser will not be directly part of the evaluation, it must nonetheless be tuned so that it can participate in the second stage tuning of the lexicalized parsing model.

In the second stage, I choose the following parameter values, using the best parameters of stage one in the underlying phrase structure and dependency components of the factored system.

- For the lexicalized parser:
  - whether to train the dependency portion of the model from scratch, or from the best lexical heads model;
  - how many learning passes to make through the training data (values considered are 0, 1, 2 and 3: higher values might be more beneficial, but training is slow for the factored models.)
- For the experimental parser:
  - whether to train the dependency portion of the model from scratch, or from the best dependency model;
  - how many learning passes to make through the training data (values considered are 0, 1, 2 and 3; training is slow for the factored models.)

In the final stage, I run the following on the *test set*:

1. the best stage-one phrase structure parser;
2. the best stage-one dependency parser;
3. the best stage-two lexicalized parser; and
4. the best experimental-model parser from stage two.

I then compare the result from 1 with 4, according to the phrase structure metrics; the result from 2 with 4, according to the dependency metrics; and the result from 3 with 4, according to the phrase structure metrics.

4.8 Summary

In this chapter, I have described an experiment to test whether the new parsing model based on context-free filtering grammar can outperform its underlying phrase structure and dependency parsing components, as well as a factored lexicalized parser in the style of Klein and Manning (2002).

I discussed some of the challenges in designing this experiment: the ambiguities in the syntactic representations I have used; the lack of gold-standard dependency trees; the difficulty in measuring parser performance; and the need for custom significance tests. I described the German TüBa-D/Z treebank used with the experiment, and the preprocessing that I performed on this corpus, including the extraction of a dependency tree for each phrase structure tree in the treebank. I reviewed briefly some of the more important decisions made while implementing the parsing system used in the experiment, and explained the means by which I generated grammars for the parser.

Finally, I described how the experiment would be run:

1. I would begin with some preliminary tests to tune a number of parameters, on a development set drawn from the German TüBa-D/Z treebank;

2. I would then run tuned versions of a pure phrase structure parser, a pure dependency parser, a factored parser in the style of Klein and Manning (2002), and a factored parser with the experimental model, on a test set drawn from the German TüBa-D/Z treebank;
3. I would determine whether the experimental model outperforms the other three.

The results of this experiment are presented in the following chapter.

Chapter 5

Experimental Results

5.1 Overview

The previous chapter described in detail the design of the experiment and the corpus of German parse trees used. This chapter is a report on the results and an analysis thereof.

Anyone considering taking seriously the significance values reported in this chapter is cautioned to first read section 4.3.4 (Establishing Statistical Significance), because the assumptions of the test used are not entirely justifiable.

5.2 Review of Hypotheses

The parsing system outlined in the previous chapter aims to combine the strengths of context-free parsers and dependency parsers. Accordingly, we would expect the following.

1. The experimental model will produce better phrase structure trees than a well-tuned unlexicalized PCFG parser.

2. The experimental model will produce better dependency trees than a well-tuned dependency parser.

3. The experimental model will produce better phrase structure trees than a well-tuned (but context-free) lexicalized parser.

In each case, the null hypothesis is that the experimental model will either under-perform the other parser, or that any improvement shown by the experimental model can likely be attributed to chance. Although demonstrating that the null hypothesis can likely be rejected is arguably due diligence in any experiment, in this case we are more interested in being able to conclude that the new system is better than existing systems to a degree large enough to justify its additional complexity.

5.3 Review of Method

The first stage of the experiment is to determine good values for the parameters left unspecified for the basic parsers in section 3.4. The second stage is to determine good values for the parameters used in the two factored parsers. The third stage, the only stage to be performed on the test set, is to generate results for all of the tuned parsers and to compare them.

For tuning purposes, I run a parser with a range of parameter values, and choose the parameter value for which the parser achieves the highest perfect score on the development data. I choose the perfect score for its simplicity and because during development it showed a relatively large variability across parsers.

All stages of the experiment use exclusively the TüBa-D/Z corpus, and the dependency trees heuristically extracted from it, as described in section 4.4.3. This corpus is first split into a training set, a development set and an test set, as described in section 4.4.2.

In the first stage, I choose the following parameter values.

- For the phrase structure parser:

  - the amount of vertical context to use in rule generation (values considered are none, parent encoding and grandparent encoding);
  - the amount of horizontal context to use in rule generation (values considered are 0, 1, 2, 3 and infinity.)

- For the dependency parser:
  - the minimum number of times a feature must occur in training data before it is included in the model (values considered are 1, 5 and 10);
  - the number of learning passes through the training data (values considered are 1 through 8.)

- For the lexical heads parser:
  - the minimum number of times a feature must occur in training data before it is included in the model (values considered are 1, 5 and 10);
  - the number of learning passes through the training data (values considered are 1 through 8.)

Although the lexical heads parser will not be directly part of the evaluation, it must nonetheless be tuned so that it can participate in second stage tuning of the lexicalized parsing model.

In the second stage, I choose the following parameter values, using the best parameters of stage one in the underlying phrase structure and dependency components of the factored system.

- For the lexicalized parser:
  - whether to train the dependency portion of the model from scratch, or from the best lexical heads model;

    - how many learning passes to make through the training data (values considered are 0, 1, 2 and 3: higher values might be more beneficial, but training is slow for the factored models.)

- For the experimental parser:

    - whether to train the dependency portion of the model from scratch, or from the best dependency model;

    - how many learning passes to make through the training data (values considered are 0, 1, 2 and 3, training is slow for the factored models.)

In the final stage, I run the following on the *test set*:

1. the best stage-one phrase structure parser;
2. the best stage-one dependency parser;
3. the best stage-two lexicalized parser; and
4. the best experimental-model parser from stage two.

I then compare the result from 1 with 4, according to the phrase structure metrics; the result from 2 with 4, according to the dependency metrics; and the result from 3 with 4, according to the phrase structure metrics.

5.4 Parser Tuning

I first describe the results of the first two tuning stages. Although only the values of the perfect metric were used to select parameter values, I report the other metrics for completeness.

horizontal context	base	parent	grandparent
0	0.213	0.339	0.365
1	0.267	0.380	0.388
2	0.269	0.388	**0.397**
3	0.266	0.390	0.396
infinity	0.275	0.382	0.394

Table 5.1: PCFG tuning: perfect scores.

horizontal context	base	parent	grandparent
0	1.000	1.000	1.000
1	1.000	1.000	1.000
2	1.000	0.999	**0.997**
3	0.999	0.999	0.997
infinity	0.999	0.999	0.997

Table 5.2: PCFG tuning: coverage of development sentences.

5.4.1 PCFG Parser

The perfect scores for all phrase structure parameter settings are shown in table 5.1. We see that the best result is achieved for grandparent encoding and two siblings worth of horizontal context. This result is consistent with the findings of Klein and Manning (2003), on the English Penn treebank. For a sense of perspective, consider that the 95% confidence interval on the perfect score of the best parser in this table is (0.368, 0.427).[1]Roughly half the results in the table fall within this interval.

All other scores are shown in tables 5.2, 5.3 and 5.4. These results compare favourably to those published by Kübler et al. (2006), who worked with an earlier (smaller) version of the TüBa-D/Z corpus.

5.4.2 Dependency Parser

The perfect scores for all dependency parameter settings are shown in table 5.5. We see that the best result is achieved for 5 training passes using all features seen at least once

[1]The perfect score is treated here as a binomially distributed variable, and is approximated as normally distributed.

context	**crossing brackets**	**base**	**parent**	**grandparent**
0	precision	0.851	0.887	0.898
	recall	0.812	0.881	0.889
1	precision	0.887	0.911	0.914
	recall	0.854	0.898	0.905
2	precision	0.888	0.914	**0.916**
	recall	0.854	0.900	**0.904**
3	precision	0.887	0.914	0.915
	recall	0.854	0.901	0.903
infinity	precision	0.886	0.909	0.911
	recall	0.851	0.895	0.899

Table 5.3: PCFG tuning: micro-averaged labeled constituent precision and recall.

context	**crossing brackets**	**base**	**parent**	**grandparent**
0	average	1.31	1.03	0.98
	sents without	0.525	0.583	0.590
1	average	1.02	0.88	0.85
	sents without	0.590	0.616	0.627
2	average	1.00	0.85	**0.83**
	sents without	0.587	0.625	**0.628**
3	average	1.00	0.84	0.85
	sents without	0.579	0.628	0.627
infinity	average	1.02	0.90	0.90
	sents without	0.580	0.614	0.613

Table 5.4: PCFG tuning: mean crossing brackets per (parsed) sentence, and proportion of (parsed) sentences without any crossing brackets.

passes	1+ instances	5+ instances	10+ instances
1	0.337	0.325	0.313
2	0.357	0.336	0.321
3	0.369	0.337	0.326
4	0.372	0.342	0.328
5	**0.375**	0.346	0.336
6	0.371	0.346	0.333
7	0.367	0.342	0.332
8	0.368	0.343	0.330

Table 5.5: Dependency tuning: perfect scores by number of training passes and minimum number of feature instances in training data.

in the training data. For a sense of perspective, consider that the 95% confidence interval on the perfect score of the best parser in this table is (0.346,0.404).[2] Most of the results over the same set of features fall within this interval.

All other dependency scores are shown in table 5.6.

The perfect scores for all lexical heads parameter settings are shown in table 5.7. A few things are worth noting. Firstly the scores are considerably higher than they are for the dependency parser. This suggests that the task is quite a bit easier. While it is true that the lexical heads parser does not need to generate non-projective trees, the size of the difference is still surprising. Secondly, there does not seem to be any pattern relating the number of training passes to the accuracy of the resulting model. This is a concern, but I nonetheless carry through with the original method of simply choosing the parameter-set with the best results. The best combination is to train for 8 passes using all features seen at least once in the training data.

All other lexical heads scores are shown in table 5.8.

Both versions of the dependency parser perform best with the maximum number of features. This is perhaps surprising, because of the potential for over-fitting if rare features are too highly weighted in the model. On the other hand, the weighting of rare

[2]The perfect score is treated here as a binomially distributed variable, and is approximated as normally distributed.

	passes	**1+ instances**	**5+ instances**	**10+ instances**
1	precision	0.879	0.871	0.864
	recall	0.880	0.874	0.870
2	precision	0.883	0.876	0.871
	recall	0.883	0.879	0.874
3	precision	0.886	0.877	0.871
	recall	0.886	0.880	0.874
4	precision	0.887	0.877	0.871
	recall	0.885	0.880	0.874
5	precision	**0.887**	0.877	0.871
	recall	**0.885**	0.879	0.874
6	precision	0.887	0.876	0.870
	recall	0.885	0.879	0.873
7	precision	0.887	0.876	0.870
	recall	0.885	0.878	0.872
8	precision	0.888	0.875	0.870
	recall	0.885	0.878	0.873

Table 5.6: Dependency tuning: non-root governor precision and recall.

features may be entirely irrelevant if these hardly ever occur in the test set. If this is in fact the case, then the extra features may be improving the score not by being relevant, but rather by serving as 'slack' variables that absorb some of the noise in the training corpus.

passes	**1+ instances**	**5+ instances**	**10+ instances**
1	0.422	0.417	0.401
2	0.437	0.423	0.405
3	0.441	0.417	0.409
4	0.447	0.416	0.415
5	0.452	0.415	0.415
6	0.446	0.414	0.408
7	0.455	0.415	0.411
8	**0.457**	0.418	0.407

Table 5.7: Lexical-heads tuning: perfect scores by number of training passes and minimum number of feature instances in training data.

	passes	1+ instances	5+ instances	10+ instances
1	precision	0.880	0.875	0.871
	recall	0.884	0.880	0.876
2	precision	0.887	0.879	0.875
	recall	0.889	0.883	0.880
3	precision	0.889	0.880	0.875
	recall	0.890	0.884	0.879
4	precision	0.890	0.880	0.876
	recall	0.890	0.885	0.880
5	precision	0.889	0.880	0.875
	recall	0.880	0.884	0.879
6	precision	0.890	0.880	0.874
	recall	0.889	0.884	0.878
7	precision	0.891	0.880	0.875
	recall	0.890	0.884	0.878
8	precision	**0.892**	0.880	0.874
	recall	**0.890**	0.884	0.878

Table 5.8: Lexical-heads tuning: non-root governor precision and recall.

parser	perfect phrase	coverage
trained init, 0 mixed passes	0.382	0.996
trained init, 1 mixed passes	0.379	0.995
trained init, 2 mixed passes	**0.388**	**0.996**
trained init, 3 mixed passes	0.376	0.996
clean init, 1 mixed pass	0.375	0.995
clean init, 2 mixed pass	0.387	0.994
clean init, 3 mixed pass	0.388	0.994

Table 5.9: Lexicalized parser tuning: perfect scores by number of training passes and lexical heads model initialization.

5.4.3 Combined PCFG-Dependency Parser

In the second tuning phase, I combine the best of the dependency parsers with the best phrase structure parser. The perfect scores for the lexicalized model are shown in table 5.9, and for the experimental model are shown in table 5.10. Additional phrase structure metrics are shown in table 5.11, and additional dependency metrics, in table 5.12.

The phase 2 tuning results are troubling. From the point of view of the experimental hypothesis, the experimental model is not obviously doing significantly better than the

parser	**perfect phrase**	**perfect dependency**	**coverage**
trained init, 0 mixed passes	0.389	0.387	0.997
trained init, 1 mixed passes	0.390	0.386	0.997
trained init, 2 mixed passes	0.396	0.394	0.997
trained init, 3 mixed passes	**0.398**	**0.393**	**0.996**
clean init, 1 mixed pass	0.383	0.360	0.996
clean init, 2 mixed pass	0.393	0.379	0.994
clean init, 3 mixed pass	0.394	0.375	0.992

Table 5.10: New model parser tuning: perfect scores by number of training passes and dependency model initialization.

parser	**precision**	**recall**	**x-brackets**	**0 x-brackets**
lexicalized				
trained init, 0 mixed passes	0.914	0.904	0.822	0.638
trained init, 1 mixed passes	0.914	0.901	0.801	0.649
trained init, 2 mixed passes	**0.914**	**0.903**	**0.792**	**0.647**
trained init, 3 mixed passes	0.910	0.901	0.831	0.639
clean init, 1 mixed pass	0.914	0.901	0.799	0.645
clean init, 2 mixed pass	0.913	0.898	0.823	0.650
clean init, 3 mixed pass	0.914	0.901	0.805	0.659
experimental model				
trained init, 0 mixed passes	0.914	0.904	0.842	0.640
trained init, 1 mixed passes	0.915	0.904	0.815	0.649
trained init, 2 mixed passes	0.916	0.906	0.802	0.655
trained init, 3 mixed passes	**0.916**	**0.905**	**0.796**	**0.655**
clean init, 1 mixed pass	0.912	0.900	0.864	0.640
clean init, 2 mixed pass	0.913	0.897	0.833	0.647
clean init, 3 mixed pass	0.914	0.894	0.827	0.651

Table 5.11: Mixed parser tuning: labeled constituent precision and recall, mean crossing brackets per sentence and proportion of sentences with no crossing brackets.

parser	**precision**	**recall**
trained init, 0 mixed passes	0.889	0.879
trained init, 1 mixed passes	0.884	0.881
trained init, 2 mixed passes	0.884	0.882
trained init, 3 mixed passes	**0.884**	**0.881**
clean init, 1 mixed pass	0.872	0.868
clean init, 2 mixed pass	0.877	0.869
clean init, 3 mixed pass	0.879	0.867

Table 5.12: New model tuning: Non-root governor precision and recall.

competition. From the point of view of tuning, different configurations are winning on different metrics (for instance, the best lexicalized configuration for mean crossing brackets is not the best lexicalized configuration for 'proportion of sentences with 0 crossing brackets', table 5.11.) Since I choose the best configuration based solely on the perfect score, it will likely be the case that significant differences will only be meaningful for comparisons of perfect score. Had I tuned for different metrics, I might have seen different final results.

In terms of perfect score, the best lexicalized configuration initializes the dependency model to the best weights from phase one (8 training passes on all features) and applies the mixed training for 2 additional passes.[3] The best experimental model configuration initializes the dependency model weights from phase one (5 training passes on all features) and applies the mixed training for 3 additional passes.

5.5 Experimental Results

The final perfect results of the four tuned parsers on the test set are shown in table 5.13. The dependency results are in table 5.14 and the phrase structure results in table 5.15.

The experimental model does not significantly outperform the best dependency parser, nor does it outperform the lexicalized parser. The experimental model does improve significantly ($p < .01$) and substantially over the PCFG parser with respect to the two crossing brackets measures. The lexicalized parser does similarly well on crossing brackets (there is no significant difference between the lexicalized and the experimental parser on these measures.)

Of interest is the failure of either the experimental model or the lexicalized model to noticeably outperform the unlexicalized PCFG parser, except on the crossing brackets

[3]A review of the results on development data, after the completion of the experiment, revealed that the 2 additional passes of mixed training had a *very* detrimental impact on the dependency model of the lexicalized parser. The precision and recall dropped to roughly 70%, down from over 88% before the mixed training.

parser	dependency perfect	phrase perfect	phrase coverage
best PCFG parser	–	0.361	0.998
best dependency parser	0.355	–	–
best lexicalized parser	–	0.359	0.998
best experimental parser	0.366	0.359	0.998

Table 5.13: Final results: proportion of completely correct parses, and coverage.

parser	precision	recall
best dependency parser	0.885	0.884
best experimental parser	0.883	0.883

Table 5.14: Final results: non-root governor precision and recall.

measures.

5.6 Detailed Comparison of Results

Aspects of the final results are not surprising. That the lexicalized parser and the experimental parser should have very similar results is probably because both use almost exactly the same lexical dependency structure within phrases below the field/clause level (the only difference being in handling of co-ordination.)

The disappointing result that neither the lexicalized parser nor the experimental parser are able to outperform an unlexicalized PCFG parser on labeled constituent precision and recall is consistent with some previous findings. Gildea (2001) finds bilexical dependencies (in the model 1 of Collins, on the English Penn treebank) to be of very little import, and Dubey and Keller (2003) (also, Dubey, 2005) find lexical head-modifier dependencies to be unhelpful on the German NEGRA treebank. Kübler et al. (2006) do

parser	precision	recall	x-brackets	0 x-brackets
best PCFG parser	0.909	0.900	0.963	0.590
best lexicalized parser	0.912	0.906	0.832	0.624
best experimental parser	0.912	0.906	0.851	0.632

Table 5.15: Final results: labeled constituent precision and recall, mean crossing brackets per sentence and proportion of sentences with no crossing brackets.

	PCFG parser	**lexicalized**	**experimental**
unlabeled precision	0.925	0.932	0.932
unlabeled recall	0.916	0.926	0.926
labeled precision	0.909	0.912	0.912
labeled recall	0.900	0.906	0.906

Table 5.16: Comparison of labeled and unlabeled constituent results on test set data.

find lexicalization to be useful with TüBa-D/Z , but only on *labeled* constituent precision and recall, not on unlabeled precision and recall.[4] Table 5.16 tells a different story for our experiment: here unlabeled scores improve more noticeably than the labeled scores. This may highlight a weakness of the model I have used, namely that the lexical head of each constituent is not directly tied to constituent labeling.

The improvement in the crossing-brackets metrics, seen with the experimental and the lexicalized model, is interesting. This suggests that the factored lexicalized models generate parse trees that have better 'shape', but that this is not reflected in some of the metrics. It is difficult to find comparable suggestions in the literature: constituent precision and recall are almost always reported in statistical parsing publications, but crossing-brackets measures are frequently omitted. Dubey and Keller (2003) report an experiment comparing an unlexicalized PCFG model, a lexicalized PCFG model and an implementation of the Collins model 1 in which the Collins model has a similar advantage in avoiding crossing brackets on German NEGRA trees, but the cause is unclear.

A further question we can ask about the parsing results is which structures are easiest to parse. There are a number of things to note in table 5.17. Firstly, with all of the parsers, some constituents are extremely easy to identify: the finite and non-finite verb 'phrases' (which are almost always unit constituents); the sentence bracketing fields (LK and VC); and the field containing complementizers in subordinate clauses (C.) Generally

[4]Moreover, the best result they report is with an *unlexicalized* run of the Stanford parser using parent encoding and rule Markovization. It would be interesting to see if lexicalization could also improve *that* parser configuration, or if lexicalization is only effective when combined with a weaker unlexicalized model.

the common fields (VF,LK,MF,VC) and phrases (ADJX,NX,ADVX,VXINF,VXFIN) are relatively easy to spot. Prepositional phrases (PX), apparently, are somewhat more difficult to identify, which is a bit puzzling. Also surprising is that the final field (NF) is difficult to find. Although the Nachfeld is not as common as the other main-clause fields, we might expect a model to consider very few competing hypotheses to cover a span following the (very identifiable) VC of a clause.

For the most part, the lexicalized and unlexicalized parsers display the same pattern of success and failure with respect to different kinds of constituents. The differences that do exist are mostly on relatively infrequent constructions, where chance probably plays a substantial role. Of some concern, however, is the degradation of performance by the more complex models on co-ordinated fields (FKONJ, the fields being co-ordinated; FKOORD, the result of the co-ordination.) The decline in performance on EN-ADD constituents is less alarming: arguably these labels should all have been replaced by NX prior to training and testing.

The poor performance on co-ordinated fields by the factored models raises the question of whether these models are failing on *all* co-ordinated structures, or just on co-ordinated fields. Most co-ordinated structures are not distinguished by their node labels, so there is nothing in the output of the parsers that allows us to identify them. Calculating co-ordination precision is therefore impossible. The gold-standard parses, however, *do* distinguish co-ordinated structures: the edge-labels of conjuncts have the value KONJ. Table 5.18 shows the recall of conjuncts, and the recall of parents of conjuncts, as produced by the three parsers that can generate phrase structure. If anything, we note a modest improvement with the factored models, so it would appear that parsing of co-ordination in general has not been compromised by lexicalization.

Another concern that we might have is whether or not the dependency trees produced by the experimental model are usable. Although the accuracies over all dependencies are the same with the pure dependency parser and the experimental model parser, is

label	description*	count	PCFG parser	lexicalized	experimental
NX	noun phrase	6855	(0.891/0.869)	(0.912/0.878)	(0.911/0.876)
PX	prepositional phrase	1825	(0.815/0.808)	(0.835/0.824)	(0.840/0.828)
ADJX	adjectival phrase	1632	(0.913/0.912)	(0.942/0.928)	(0.930/0.922)
VXFIN	finite verb phrase	1496	(1.000/0.999)	(1.000/0.999)	(1.000/0.999)
ADVX	adverbial phrase	1161	(0.963/0.948)	(0.956/0.942)	(0.961/0.945)
VXINF	non-finite verb phrase	735	(0.999/0.997)	(0.999/0.997)	(0.999/0.997)
EN-ADD	proper noun or named entity	307	(0.793/0.687)	(0.712/0.700)	(0.699/0.704)
FX	foreign language phrase	33	(0.897/0.788)	(0.900/0.818)	(0.900/0.818)
DP	determiner phrase	13	(0.429/0.692)	(0.700/0.538)	(0.900/0.692)
MF	Mittelfeld	1577	(0.912/0.916)	(0.893/0.921)	(0.899/0.923)
LK	Linke (Satz-)Klammer	1136	(0.994/0.996)	(0.994/0.994)	(0.994/0.993)
VF	Vorfeld	1072	(0.950/0.951)	(0.936/0.949)	(0.935/0.949)
VC	Verbkomplex	964	(0.987/0.989)	(0.987/0.990)	(0.986/0.988)
C	complementizer field	368	(0.989/0.978)	(0.986/0.976)	(0.986/0.978)
NF	Nachfeld	351	(0.770/0.783)	(0.738/0.795)	(0.738/0.882)
FKONJ	conjunct containing fields	151	(0.662/0.662)	(0.606/0.602)	(0.573/0.649)
FKOORD	coordinated fields	79	(0.553/0.532)	(0.455/0.506)	(0.440/0.506)
KOORD	field for coordinating particles	53	(0.712/0.906)	(0.783/0.887)	(0.774/0.906)
PARORD	field for non-coord particles	13	(0.000/0.000)	(0.500/0.077)	(0.500/0.077)
LV	resumptive construction	9	(0.500/0.667)	(0.455/0.556)	(0.364/0.444)
VCE	Verbkomplex with the split finite verb of Ersatzinfinitiv constructions	1	(1.000/1.000)	(1.000/1.000)	(1.000/1.000)
SIMPX	simplex clause	1494	(0.874/0.892)	(0.856/0.897)	(0.862/0.898)
R-SIMPX	relative clause	135	(0.871/0.852)	(0.892/0.859)	(0.906/0.859)
DM	discourse marker	17	(0.688/0.647)	(0.667/0.588)	(0.786/0.647)
P-SIMPX	paratactic construction of simplex clauses	6	(0.333/0.167)	(0.250/0.167)	(0.250/0.167)

Table 5.17: Breakdown of constituent (precision/recall) by constituent label. *descriptions adapted (shortened) from Telljohann et al. (2005).

	PCFG parser	lexicalized	experimental
conjunct recall	0.747	0.763	0.750
parent of conjunct recall	0.629	0.645	0.655

Table 5.18: Comparison of co-ordination accuracies on test set data.

	pure dependency parser	**experimental parser**
clause-level precision	0.852	0.849
clause-level recall	0.854	0.854
overall precision	0.885	0.883
overall recall	0.884	0.883

Table 5.19: Comparison of clause-level dependency precision and recall, where 'clause-level' means either the governor or dependent token is a verb.

it possible that the two models are getting different aspects of the parses correct? In particular, the context-free filtering grammar used by the experimental model enforces a tight correspondence between dependency and phrase structure *within* arguments and adjuncts, but allows a great deal more flexibility at the level of verb-argument structure. Is it possible that the experimental model is doing much better at one of these levels than the other? To examine this possibility, we can compute the dependency precision and recall of relations in which the governing or dependent token is a verb: this essentially amounts to looking only at the accuracy of chains of verbs, of attachments of arguments and adjuncts to verbs, and of attachments of subordinate clauses into their parent clauses. We can see from table 5.19 that although the accuracies of this restricted group of dependencies are lower than accuracies in general, the experimental model parser and the pure dependency parser perform almost identically on both metrics. The filtering grammar has not damaged the experimental model's ability to identify dependencies at the clause level.

As a final sanity check, how do the parsers of this experiment compare with similar studies on this corpus? Unfortunately, there have been almost no publications of full statistical parsing work on the TüBa-D/Z corpus. The most comparable paper of which I am aware is that of Kübler et al. (2006). Even with this paper, a direct comparison can be misleading: the available training corpus sizes are different, the evaluation sampling methods differ and it is not clear whether the gold standard trees are the same. Kübler et al. use an earlier version of the corpus, one which only has 15,000 sentences; our

	Kübler (2006)	**PCFG parser**	**experimental parser**
unlabeled precision	0.923	0.925	0.932
unlabeled recall	0.909	0.916	0.926
labeled precision	0.899	0.909	0.912
labeled recall	0.885	0.900	0.906

Table 5.20: Comparison of results with Kübler (2006).

version has 22,000. Kübler et al. report results of 10-fold cross-validation over the whole corpus, using all sentences with 40 or fewer *words*; I select separate training, development and test sentences only once, at random, and use all sentences with 40 or fewer *tokens* (including punctuation.) Kübler et al. do not state how they deal with disconnected subtrees in TüBa-D/Z , and this may impact constituent precision and recall.

Both studies provide gold standard part of speech tags as input, and Kübler et al. use the Stanford parser, which instantiates many of the same statistical parsing techniques behind my PCFG parser and lexicalized parser.

Table 5.20 compares the best results of Kübler et al. (2006) with my results, according to precision and recall on labeled and unlabeled constituents. Comparing against the *best* results of Kübler et al. is not entirely fair to my parser, since choosing one parser configuration from among several is a form of tuning that ideally ought to be validated by a further run against a test set.[5] On the other hand, my results are also likely to be inflated by my drawing of samples at uniform from the entire corpus, by my shorter sentence length restrictions, and by my larger training set.

The absolute scores of the phrase structure parsers in this experiment do not look unreasonable.

The dependency results cannot be compared with other studies, because the heuristically extracted trees have only ever been used for this experiment.

[5]Note, however, that Kübler et al. do not themselves do any tuning: they use parsers out-of-the-box that have been developed by others, on a different corpus. Since they do no tuning and jump directly to evaluating hypotheses about the relative strengths of their parsers, they can treat the entire corpus as unseen data and their use of cross-validation is justified.

5.7 Conclusion

The experimental hypothesis, that the experimental model would improve upon pure phrase structure and pure dependency analysis, is not supported by the evaluation. The experimental model does, at least, hold its own.

5.8 Summary

We have reviewed an experiment designed to demonstrate that the constraints between dependency structure and phrase structure, imposed by context-free filtering grammar, can improve phrase structure and dependency parsing accuracy beyond what is possible with specialized phrase structure or dependency parsers. A experimental parser, based on the models and algorithms of the previous chapter, was compared on German TüBa-D/Z data against a pure PCFG parser, a pure dependency parser (in the style of McDonald et al., 2005b) and a factored lexicalized phrase structure parser. All parsers were tuned on development data, then compared on test set data, where few significant differences were found between the performances of any of the parsers.

Although the result of the experiment is disappointing, the experimental model at least did not underperform the other models. Since the experimental model permits a co-ordinated analysis of phrase structure and dependency structure, it may still be useful in contexts where a context-free phrase structure analysis is inadequate (e.g. in applications relating to freer-word-order languages) but a pure dependency analysis is undesirable (e.g. because a commitment has been made to a constituent-based semantic analysis of sentences.)

Chapter 6

Conclusion

As we have seen in Chapter 2, the phrase structure analysis of freer-word-order languages is problematic. Early statistical parsing work on German (Dubey and Keller, 2003; Dubey, 2005; Schiehlen, 2004), relying on the NEGRA treebank (Skut et al., 1997), did not show very promising results. More recent work (Kübler, 2005; Kübler et al., 2006), on the TüBa-D/Z treebank (Telljohann et al., 2005), has shown much better results, but the TüBa-D/Z corpus does not encode verb-argument structure in a way that can be accurately modeled by existing PCFG-based parsers.

On the other hand, statistical dependency parsers, long ago left in the dust by CFG-based parsers, are making a comeback (Nivre et al., 2006; McDonald and Pereira, 2006; Yamada and Matsumoto, 2003; Buchholz and Marsi, 2006). The newest of these parsers can handle with relative ease the non-projective dependencies characteristic of freer-word-order languages.

This thesis aims to provide a statistical method for the comprehensive syntactic analysis of freer-word-order languages by co-ordinating phrase structure parsing with dependency parsing. The foundation of this analysis (Chapter 3) is *context-free filtering grammar*, a generalization of context-free grammar, inspired by the factored parsing model of Klein and Manning (2002). Context-free filtering grammar specifies for each constituent, through annotated production rules, which tokens within the constituent can take part in dependency relations with tokens outside the constituent. I provide a chart-parsing algorithm to recover all of the syntactic structures matching a given sentence, a statistical parametrization to resolve structural ambiguities, and scoring heuristics that allow A* search to be used to find the model-optimal parse for a sentence.

Chapter 4 describes an experiment to test the hypothesis that the new parser can outperform other parsers that do not use the new grammar formalism. The competing parsers are an unlexicalized PCFG parser, a maximum-spanning-tree dependency parser and a factored parser using a grammar restricted to one exposed token per constituent. All parsers are tuned on development data and then tested on evaluation data.

All of the parsers turn out to have similar performance (Chapter 5), with the two factored parsers (the restricted model and the new model) beating the PCFG parser only on crossing bracket measures. The only positive conclusion that can be drawn from the experiment is that the new model does not lag behind the others.

6.1 Summary of Contributions

context-free filtering grammar: The new grammar formalism provides a framework in which to analyze the interaction between phrase structure and dependency structure. It may be of use on its own or, more likely, as an inspiration for more nuanced formalisms.

a parsing algorithm for context-free filtering grammar: The existence of a chart-parsing algorithm for the new type of grammar makes the grammar more accessible to the research community than it might otherwise be, and permits an intuitive understanding of the properties of the new grammar.

an implementation of a parser using context-free filtering grammar: I implemented the parsing system described in this thesis, demonstrating its feasibility.

a TüBa-D/Z -derived dependency treebank: The heuristically extracted dependency treebank used for my experiment could be an excellent starting point for a manually annotated/verified dependency treebank of German.

an evaluation of a context-free filtering grammar model: The results of Chapter 5 demonstrate that the new parsing model is capable of generating phrase structure and dependency analyses that are (independently of one another) just as good as those produced by pure phrase structure or pure dependency parsers.

6.2 Limitations

the new parser combines heterogeneous models: The parser implemented with this thesis combines a Bayesian probabilistic model with a linear large-margin model. The result, while practically effective, has unclear mathematical properties. The use of a heterogeneous factoring may explain why the factored models fail to outperform the base models in my experiment.

unlabeled dependency parsing: Unlabeled dependency parsing, while challenging, is not particularly useful on its own. Unlabeled dependency trees encode syntactic relations very ambiguously, and are not adequate as input to a (hypothetical) semantic analysis process. McDonald et al. (2006) adds labels to dependency trees as a post-processing step, and this should be feasible with my parser as well.

unvalidated dependency treebank: The dependency treebank required by the experiment is heuristically extracted from the TüBa-D/Z corpus, but has not been validated by qualified linguists. Better results may be possible with a more consistently correct dependency treebank.

no error analysis: No attempt was made in this work to differentiate between errors made because of flaws in the models, because of pruning and because of inconsistencies in the corpus annotation.

6.3 Future Work

fully large-margin factored model: An exciting project is to apply the online large-margin learning method used by McDonald et al. (2005a) to a phrase structure parser. Taskar et al. (2004) have already shown that features can be chosen for such a model, but they did not use a computationally efficient learning approach, such as that of McDonald et al. (2005a), to learn the model weights. With a linear large-margin model for both phrase structure and dependency structure, a homogeneous factored model would be immediately feasible. The training of such a model would be far less awkward than that of the model used in this thesis.

tecto-grammatical phrase structure analysis: An inspiration for this thesis was the paper of Penn and Haji-Abdolhosseini (2003), which aimed to use topological field analysis as a guide for a semantically motivated phrase structure analysis of freer-word-order languages. The problem addressed in that paper was to efficiently search for the correct phrase structure without making the rigid ordering and contiguity assumptions of CFG. The problem of how to determine the constraints on ordering and contiguity that ought to exist in a freer-word-order grammar remains unsolved, however. A labeled dependency analysis combined with a phrase structure analysis (produced by a filtering grammar parser) in the TüBa-D/Z style could provide 'hints' to a tecto-grammatical parser as to which parse derivations are worth exploring for a sentence.

eliminating the need for pruning: Investigation subsequent to the experiment in this thesis suggests that the pruning of arcs during A* search is having a significant impact on parsing accuracy. It would be beneficial to reduce the memory requirements of the parser, both by optimizing the CKY outside-score precomputation and by using a tighter dependency scoring heuristic.

extending the error analysis: It would be valuable to differentiate between errors made because of flaws in the models, because of pruning and because of inconsistencies in the corpus annotation.

Bibliography

Anne Abeillé, Yves Schabes, and Aravind K. Joshi. 1990. Using lexicalized TAGs for machine translation. In *Proceedings of the 13th conference on Computational linguistics*, pages 1–6, Helsinki, Finland. Association for Computational Linguistics.

Steven P. Abney. 1997. Stochastic attribute-value grammars. *Computational Linguistics*, 23(4):597–618.

James Allen. 1994. *Natural language understanding*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, second edition.

Hiyan Alshawi, Shona Douglas, and Srinivas Bangalore. 2000. Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26(1):45–60.

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

E. Black, S. Abney, S. Flickenger, C. Gdaniec, C. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. Procedure for quantitatively comparing the syntactic coverage of English grammars. In *Human Language Technology '91: Proceedings of the workshop on Speech and Natural Language*, pages 306–311, Pacific Grove, California. Association for Computational Linguistics.

Ezra Black, Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer, and Salim Roukos. 1993. Towards history-based grammars: using richer models for probabilistic parsing. In *Proceedings of the 31st annual meeting of the Association for Computational Linguistics*, pages 31–37, Columbus, Ohio. Association for Computational Linguistics.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Natural Language Learning*, New York, NY.

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

Eugene Charniak. 1996. Tree-bank grammars. In *Proceedings of the 8th annual conference on Innovative Applications of Artificial Intelligence, Vol. 2*, pages 1031–1036, Portland, Oregon. American Association for Artificial Intelligence.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 9th annual conference on Innovative Applications of Artificial Intelligence*, pages 598–603, Providence, Rhode Island. American Association for Artificial Intelligence.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the first conference of the North American chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, Washington. Morgan Kaufmann Publishers Inc.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 263–270, Ann Arbor, Michigan. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th annual meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain. Association for Computational Linguistics.

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

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Michael Collins, Lance Ramshaw, Jan Hajič, and Christoph Tillmann. 1999. A statistical parser for Czech. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 505–512, College Park, Maryland. Association for Computational Linguistics.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, page 111, Barcelona, Spain. Association for Computational Linguistics.

Koby Crammer, Ofer Dekel, Yoram Singer, and Shai Shalev-Shwartz. 2004. Online passive-aggressive algorithms. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 169–176, Bonn, Germany. ACM Press.

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

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 541–548, Ann Arbor, Michigan. Association for Computational Linguistics.

Amit Dubey. 2005. What to do when lexicalization fails: parsing German with suffix analysis and smoothing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 314–321, Ann Arbor, Michigan. Association for Computational Linguistics.

Amit Dubey and Frank Keller. 2003. Probabilistic parsing for German using sister-head dependencies. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 96–103, Sapporo, Japan. Association for Computational Linguistics.

Denys Duchier. 1999. Axiomatizing dependency parsing using set constraints. In *Sixth Meeting on Mathematics of Language*, pages 115–126, Orlando, Florida, July.

Denys Duchier and Ralph Debusmann. 2001. Topological dependency trees: a constraint-based account of linear precedence. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 180–187, Toulouse, France. Association for Computational Linguistics.

Jason Eisner. 1996a. An empirical comparison of probability models for dependency grammar. Technical report, Univ. of Pennsylvania. Technical Report IRCS-96-11.

Jason Eisner. 1996b. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th conference on Computational linguistics*, pages 340–345, Copenhagen, Denmark. Association for Computational Linguistics.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 205–208, Sapporo, Japan. Association for Computational Linguistics.

Sisay Fissaha, Daniel Olejnik, Ralf Kornberger, Karin Müller, and Detlef Prescher. 2003. *Experiments in German Parsing*, pages 50–57. Springer Berlin.

Daniel Gildea. 2001. Corpus variation and parser performance. In Lillian Lee and Donna Harman, editors, *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202.

Phillip Good. 2000. *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses, 2nd Ed.* Springer, New York, NY.

F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, A. Ratnaparkhi, and S. Roukos. 1994. Decision tree parsing using a hidden derivation model. In *Proceedings of the workshop on Human Language Technology*, pages 272–277, Plainsboro, NJ. Association for Computational Linguistics.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Mark Johnson. 2001. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association* *for Computational Linguistics*, pages 136–143, Philadelphia, Pennsylvania. Association for Computational Linguistics.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 535–541, College Park, Maryland. Association for Computational Linguistics.

Sylvain Kahane, Alexis Nasr, and Owen Rambow. 1998. Pseudo-projectivity: a polynomially parsable non-projective dependency grammar. In *Proceedings of the 17th international conference on Computational linguistics*, pages 646–652, Montreal, Quebec, Canada. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *Proceedings of the 16th annual Conference on Neural Information Processing Systems*, pages 3–10.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Morristown, NJ, USA. Association for Computational Linguistics.

Sandra Kübler. 2005. How do treebank annotation schemes influence parsing results? or how not to compare apples and oranges. In *Proceedings of the international conference on Recent Advances in Natural Language Processing*, Borovets, Bulgaria, September.

Sandra Kübler, Erhard W. Hinrichs, and Wolfgang Maier. 2006. Is it really that difficult to parse German? In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 111–119, Sydney, Australia, July. Association for Computational Linguistics.

Sandra Kübler and Jelena Prokić. 2006. Why is German dependency parsing more reliable than constituent parsing? In *Proceedings of the Fifth International Workshop on Treebanks and Linguistic Theories*, Prague, Czech Republic, December.

Roger Levy and Christopher D. Manning. 2004. 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*, page 327, Barcelona, Spain. Association for Computational Linguistics.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting of the Association for Computational Linguistics*, pages 276–283, Cambridge, Massachusetts. Association for Computational Linguistics.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, Massachusetts.

Mitchell P. Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *Proceedings of the Human Language Technology conference.*

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–98, Ann Arbor, Michigan. Association for Computational Linguistics.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Natural Language Learning*, New York, NY.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Ryan T. McDonald and Fernando C. N. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice.* State University of New York, Albany, NY.

Jens Nilsson, Joakim Nivre, and Johan Hall. 2006. Graph transformations in data-driven dependency parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 257–264, Sydney, Australia. Association for Computational Linguistics.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 149–60, Nancy, France.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 49–56. Boston, MA, USA.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data driven parsergenerator for dependency parsing. In *proceedings of the 5th international conference on Language Resources and Evaluation*, Genoa, Italy.

Gerald Penn and Mohammad Haji-Abdolhosseini. 2003. Topological parsing. In *Proceedings of the tenth conference of the European chapter of the Association for Computational Linguistics*, pages 283–290, Budapest, Hungary. Association for Computational Linguistics.

Adwait Ratnaparkhi. 1997. A linear observed time statistical parser based on maximal entropy models. In Claire Cardie and Ralph Weischedel, editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 1–10. Association for Computational Linguistics, Somerset, New Jersey.

Adwait Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Stuart Russell and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, second edition.

Michael Schiehlen. 2004. Annotation strategies for probabilistic parsing in German. In *Proceedings of the 20th international conference on Computational Linguistics*, page 390, Geneva, Switzerland. Association for Computational Linguistics.

Helmut Schmid. 2000. LoPar: design and implementation. *Arbeitspapiere des Sonderforschungsbereiches*, 340(149), July.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th conference on Computational linguistics*, pages 253–258, Helsinki, Finland. Association for Computational Linguistics.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the 5th conference on*

Applied natural language processing, pages 88–95, Washington, DC. Morgan Kaufmann Publishers Inc.

Benjamin Taskar, Dan Klein, Michael Collins, Daphne Koller, and Christopher Mannning. 2004. Max-margin parsing. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, Morristown, NJ, USA. Association for Computational Linguistics.

Heike Telljohann, Erhard W. Hinrichs, Sandra Kübler, and Heike Zinsmeister. 2005. Stylebook for the Tübingen treebank of written German (TüBa-D/Z). Technical report, Universität Tübingen, Germany. Seminar für Sprachwissenschaft.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of 8th International Workshop on Parsing Technologies*, pages 195–206.