INDUCTION OF FINE-GRAINED LEXICAL PARAMETERS OF TREEBANK PCFGS WITH INSIDE-OUTSIDE ESTIMATION AND LEXICAL TRANSFORMATIONS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Tejaswini Deoskar

August 2009

INDUCTION OF FINE-GRAINED LEXICAL PARAMETERS OF TREEBANK PCFGS WITH INSIDE-OUTSIDE ESTIMATION AND LEXICAL TRANSFORMATIONS

Tejaswini Deoskar, Ph.D.

Cornell University 2009

A probabilistic model of the structural preferences of open-class words is important for accurate parsing, irrespective of the particular parsing paradigm. However, word-specific properties are not represented adequately in statistical grammars trained solely on annotated corpora, due to the Zipfian distribution of words in a corpus. The problem becomes more severe for models containing complex, fine-grained lexical categories. This dissertation presents procedures to estimate complex lexical parameters of a smoothed Penn Treebank PCFG from unlabeled data. The PCFG contains important linguistic representations for argument-adjunct distinctions and long-distance dependencies. The lexical parameters of the PCFG encode fine-grained information about structures selected by a lexical item, such as its subcategorization frames.

Values of lexical parameters of words are re-estimated from a large source of unlabeled data using the Inside-outside algorithm for PCFG induction. Re-estimation from unlabeled data is constrained by interpolating re-estimated parameters with treebank parameters; we use the intuition that certain parameters of treebank models are more accurate than others and can guide unsupervised estimation, thus avoiding heuristic constraints. Models obtained in this way are shown to be superior to models obtained with standard Inside-outside estimation. We get substantial improvements in identification of complex subcategorized structures for unseen and low-frequency verbs in the treebank, as measured by parsing-based evaluations.

This dissertation interweaves several issues related to unsupervised estimation of PCFGs: a treebank PCFG enables evaluations of re-estimated models against a high-quality gold standard (the Penn Treebank), unlike models obtained previously by Inside-outside from completely unlabeled data. Maximum-likelihood estimation (via Inside-outside) allows examination of questions regarding the relative efficacy of supervised estimation from a treebank versus unsupervised estimation on a much larger corpus. Subcategorization frames and other features in the PCFG are based solely on Penn Treebank annotation; the fine-grained, wide-coverage lexical resource obtained here is therefore aligned with Penn Treebank structures and interpretable according to Penn Treebank annotation guidelines. The framework for creating a linguistically-sophisticated PCFG can be extended to other languages having a treebank in the Penn Treebank style.

BIOGRAPHICAL SKETCH

Tejaswini Deoskar did her schooling from Somalwar High School, Nagpur, India. She studied at the College of Engineering, Pune from 1992-1996 and obtained a degree in Electronics and Telecommunication Engineering from the University of Pune, India. She spent several years working as an electronics engineer, with the largest amount of time spent at the Inter-University Center for Astronomy and Astrophysics (IUCAA), Pune on building a digital controller and image-acquisition device for a CCD camera for the IUCAA Telescope at Giravali, Maharashtra, India. She joined the graduate program in Linguistics at Cornell University in 2002, graduating with a major degree in Linguistics and a minor degree in Cognitive Science.

ACKNOWLEDGEMENTS

I am immensely grateful to my advisor Mats Rooth for his guidance, support and feedback throughout the course of this research. His insight into the persistent problems of linguistics and computational linguistics has been truly invaluable and a source of inspiration. I am thankful to him for having his door open for countless drop-in conversations, and for occasionally having more faith in me than I had myself. Needless to say, this research would not have been possible without him.

I would like to thank the other members of my dissertation committee, Lillian Lee and John Whitman. I am grateful to Lillian for discussions and comments at different stages in this research, and for making the Cornell NLP seminar interesting and fun. John has been a constant source of encouragement throughout my graduate years at Cornell. I am also thankful to him for his guidance during my research on complex-predicates in Marathi.

It is not possible to mention all the friends and colleagues who made my life as a graduate student at Cornell not only bearable but enjoyable at most times. Dorit opened my eyes to entire new worlds – the long conversations, the walks in the Ithaca snow, and the summer swims in Cayuga lake are all unforgettable. Her love and encouragement got me through the hardest times. I thank Joe for the home-cooked dinners, Duong for the conversations in the balcony, Johanna for always being a friend, Nikola for the music, and many others for good times. I am especially grateful to the friends who introduced me to Argentine tango in my last year at Cornell and to members of the Ithaca tango community for wonderful dances – they never failed to recharge my batteries in the last year of writing this dissertation.

I am indebted to Sunu for first triggering my interest in questions related to language, and for always questioning my beliefs.

Amit has been an unwavering source of support I could always count on.

My greatest thanks go to my parents for their love and unquestioning support throughout, and to my brother for being the person he is.

I acknowledge the Cornell Theory Center, now the Cornell Center for Advanced Computing, for the use of their computing resources for the re-estimation experiments described in this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Context of this work

Parsing is the act of assigning a syntactic structure to a sentence in a natural language. A parsing component is integral to any system of natural language understanding, since access to the syntactic structure of a sentence is, to a large extent, a prerequisite for semantic processing. Numerous models for parsing natural language have been proposed in the linguistics and computational-linguistics literature, reflecting a variety of views regarding the nature and complexity of representations in the assigned syntactic structure, the computational complexity of the model, considerations of cost-effective performance on real-world tasks, etc. Modern parsing systems, like most modern computational systems, have moved away from the "knowledge-rich" paradigm of the previous decade which emphasized expensive human intervention in learning natural language systems. Currently, the emphasis is on statistical models in which the computational linguist designs an appropriately expressive model, and values of the model's parameters are estimated from natural language data, either raw sentences, or sentences with added annotation. Statistical models elegantly express the ambiguity inherent in natural language by treating parses (or other structures) as more or less likely, depending on the probability assigned to the parse. In most cases, we are interested in the most likely structure.

The most successful statistical parsing models to date have been those which are estimated from syntactically annotated data (for example, Collins (1997); Charniak (1997, 2000), amongst others). Indeed, it is the availability of large corpora of syntactically annotated sentences such as the Penn Treebank (Marcus et al., 1993) that spurred research

in statistical parsing in the last decade. Computationally, these models are stochastic or probabilistic context-free grammars. Linguistically, they are simple models of phrase-structure. They are often referred to as "treebank" grammars, since both the context-free rules in the grammar and the stochastic model are respectively obtained from the annotations and relative frequencies in a treebank. The best models trained on treebank data currently have an impressive accuracy in the low 90% range in assigning labeled brackets to constituents in an unseen sentence of English[1].

Most state-of-the-art parsing models for English, while using the Penn Treebank as their training data source, are reduced representations of the syntactic annotation present in the treebank. For example, they do not make use of the functional categories in the treebank, ignore empty categories (traces) that indicate bounded and unbounded dependencies and do not make a distinction between arguments and adjuncts (the exceptions are Collins's (1997) Model II where a distinction is made between arguments and adjuncts, and Model III where wh-movement and traces are represented). Being based on the context-free phrase structure formalism, these parsers largely model dependencies that can be expressed as local trees. The long distance dependencies involved in some linguistic phenomena, such as extraction and control, are not represented, although most NLP applications that use the output of a parser would require these in order to recover predicate-argument structures in a sentence. One reason for simplified representations in parsers is that annotated data is limited in size, leading to a severe sparseness in statistical estimates. Although the Penn Treebank is a fairly large resource for syntactically annotated data, even simple models estimated over the Penn Treebank have a large number of parameters with very sparse estimates. The problem becomes more severe for models with complex representations which have a larger number of categories, since there are more parameters to be estimated from the same amount of data. The other rea-

[1]Such a high parsing score is obtained only for within-domain sentences, and only for English. Other languages do not have parsers of such high accuracy yet.

son for the focus on simpler models in the natural language processing community is an effect of the need to emphasize quantitative evaluations. The Zipfian distribution exhibited by language data means that a model that captures the most common phenomenon exhibited in any given corpus of language data will account for a very large proportion of the data. This leads to the temptation to ignore the long tail of low frequency events such as long distance dependencies, since they have little effect on evaluation results.

More expressive grammar formalisms such as CCG (Combinatory Categorial Grammar) (Steedman, 1996, 2000) or HPSG (Head-driven Phrase Structure Grammar) (Pollard & Sag, 1994) contain representations of more complex linguistic phenomenon and can model long-distance dependencies, thus making the analyses proposed by these grammars more conducive to semantic interpretation. However, statistical models for computational grammars based on these formalisms are also more complex (Abney, 1997) and difficult to estimate. It is only recently that statistical methods have been used with these grammar formalisms, and treebank grammars become available. For example, (Hockenmaier & Steedman, 2002) constructs a CCG parser based on the Penn Treebank, (Tsuruoka & Tsujii, 2004) describes an HPSG treebank grammar and (Riezler et al., 2002) an LFG treebank grammar. Unsupervised estimation of these grammars presents an even harder problem, although, due to the severity of annotated-data sparseness for complex models, unsupervised estimation becomes more important for grammars with complex representations than for simpler grammars.

1.1.1 Lexical information and lexicalist theories

In traditional views of grammar, the lexicon is the repository of "unpredictable" word-specific information, while the grammar is the generative component, where language-

specific combinatory rules are stored. In many modern theories the division of labour between the grammar and the lexicon has moved in the direction of the lexicon. The lexicon in so-called "lexicalist" grammars is a complex repository of almost all language-specific syntax, with an impoverished syntactic component that contains only some basic principles of composition. These theories view language-specific syntax as arising mainly from the general properties of function words and the subcategorization or valence properties of content words. For example, LTAG (Joshi & Schabes, 1997), CCG (Steedman, 1996), and minimalism (Chomsky, 1995) are all lexicalist theories to varying degrees.

A probabilistic model of word-specific subcategorization properties is important for accurate parsing, irrespective of the particular parsing paradigm. However, due to the Zipfian distribution of words in a corpus of language data, information about word-specific properties is particularly difficult to obtain from a treebank corpus, which of necessity is limited in size. Consider the case of verbal subcategorization or valence. Out of all verb types (approximately 7450) in the standard training sections (0 to 22) of the Penn Treebank, about 38% occur with a token frequency of one. Another 14% occur with a token frequency of two. This means that reliable subcategorization statistics for a large number of verbs cannot be learnt from treebank data[2] – the sole source of such information is unannotated data. Addition of subcategorization information from external probabilistic wide-coverage lexicons to a treebank parser is difficult since representations in the lexicon may not match those in the treebank, with no obvious mapping between them. Wide-coverage probabilistic lexicons may not be available for most languages to begin with.

In this dissertation, we propose a system for learning complex word-specific lex-

[2]Briscoe & Carroll (1997) find that in their verbal subcategorization acquisition system, about 100 occurrences are required in order to obtain a reliable subcategorization frame distribution of a verb.

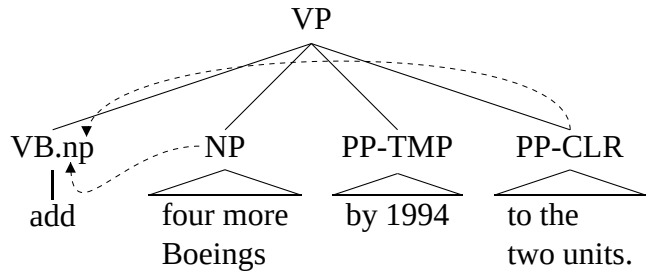
ical information from unannotated data within a parsing paradigm (via estimation of PCFG parameters using the Inside-outside algorithm, (Lari & Young, 1990)). The lexical representations learnt from unannotated data have a systematic correspondence with fine-grained annotations in a treebank, and to the lexical parameters of a treebank PCFG. Subcategorization is the typical example of such lexico-syntactic information; we may also represent other kinds of word-specific syntactic information such as the attachment preference of adverbs to sentential, nominal or verbal nodes.

1.1.2 An Unlexicalized Penn Treebank PCFG

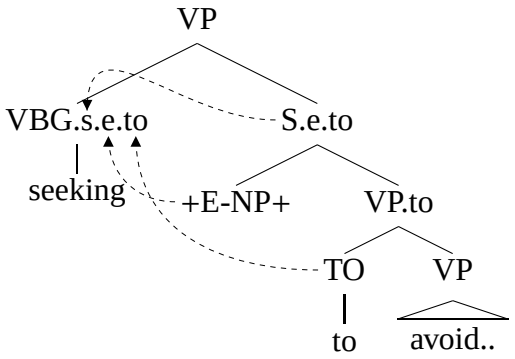
In order to have fine-grained and linguistic lexical categories (like CCG) within a simple formalism with well-understood estimation methods, we first build a PCFG containing such categories from the Penn Treebank. The PCFG is unlexicalized (with limited lexicalization of only certain function words, in the spirit of Klein & Manning (2003); Schmid (2006)). All functional tags in the PTB (such as NP-SBJ, PP-TMP, PP-CLR, etc.) are maintained, as also all empty categories, making long-distance dependencies recoverable. It is created by first transforming the Penn Treebank (Johnson, 1998) in an appropriate way and then extracting a PCFG from the transformed trees. In addition to several standard treebank transformations that are known to produce an accurate PCFG, our transformations of the treebank trees include ones in which information that is implicit in treebank structures is made explicit by marking it on treebank category symbols as features. The features are incorporated into PCFG category symbols, making them parameters of the PCFG. Through such incorporated features, the PCFG model can represent important linguistic information that is not represented overtly in most state-of-the-art PCFG parsers. The process can be repeated for other languages with a treebank annotated in the Penn Treebank style to produce a fine-grained PCFG.

We follow a general strategy of enriching lexical (pre-terminal) categories in the treebank trees by explicitly marking structural information (including long-distance structural information) on the pre-terminal category of the word that selects or *subcategorizes for* that structure. Thus, pre-terminal categories for open-class items like verbs, nouns and adverbs in the PCFG are more complex than Penn Treebank POS tags. They encode information about the structure selected by the lexical item, in effect, its subcategorization frame. A pre-terminal in the PCFG consists of the standard Penn Treebank POS tag, followed by a sequence of features incorporated into it – each Penn Treebank POS tag can be considered to be divided into multiple finer-grained "supertags" (Bangalore & Joshi, 1999; Clark & Curran, 2004) by the incorporated features. These features encode the structure selected by the words. Examples of three subcategorization frames on verbal categories, and the structures that they indicate are shown in Figure 1.1. Notice that in Figure 1.1 (a), the PP-TMP that is an adjunct is not part of the subcategorization frame marked on the verb *add*, while the two arguments NP and PP-CLR are marked as features on the verbal category. In Figure 1.1 (b), non-local information about the clausal complement of the verb *seeking*, such as the presence of an empty subject and the kind of VP within the clausal complement are both marked on the verbal category, thus making it local.

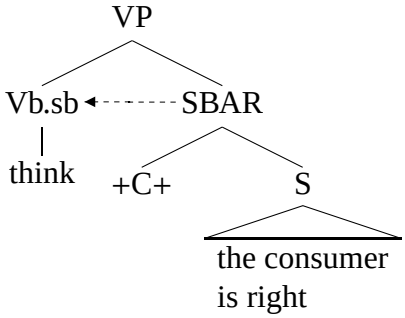
The nature of the lexical parameters of the treebank PCFG is illustrated in Figure 1.2 by the lexical entry of a commonly-occurring verb *named* in the frequency lexicon of the PCFG. Each Penn Treebank POS-tag is followed by a sequence of features (representing the subcategorization frame of the verb), followed by the frequency of the combination in the treebank (the specific feature-values are not important now, and are presented in detail in Chapter 2). It is seen that *named* occurs in the treebank with a fairly high frequency and with a variety of subcategorization frames i.e. in a variety of syntactic contexts.



(a) An NP PP subcategorization frame marked on the verb "add" as *np*. Note that the arguments NP and PP-CLR are part of the subcategorization frame and are represented locally on the verb but the adjunct PP-TMP is not.



(b) An S frame on the verb "seeking": +E-NP+ represents the empty subject of the S. Note that structure internal to S is also marked on the verb.



(c) An SBAR frame: +C+ is the empty complementizer.

Figure 1.1: Subcategorized structures are marked as features on the verbal preterminal category.

named	VBN.s.e.sc.-	118.0	VBN.n.-.-.-	20.0	VBN.np.-.-.to	15.0
	VBN.s.e.to.-	4.0	VBN.np.-.-.as	2.0	VBN.s.-.to.-	1.0
	VBN.np.-.-.for	1.0	VBD.s.-.sc.-	8.0	VBD.n.-.-.-	5.0
	VBD.np.-.-.as	4.0	VBD.np.-.-.to	1.0	VBD.s.-.to.-	2.0

Figure 1.2: Entry in the treebank PCFG lexicon for the verb *named*.

In contrast to the entry for *named*, Figure 1.3 shows lexical entries for some verbs that have a low occurrence-frequency in the treebank. The lexical entries for these verbs are quite impoverished. It is likely that these verbs do subcategorize for complements other than the ones represented in their entries here – our goal is to learn accurate distributions for these, as well as for completely unseen words, from unannotated data.

abandons	VBZ.n.-.-.-	2.0		
abate	VB.z.-.-.-	2.0		
abdicate	VBP.n.-.-.-	1.0		
abetting	VBG.p.-.-.in	1.0	VBG.s.-.to.-	1.0
abide	VB.p.-.-.by	2.0		

Figure 1.3: Some entries of verbs in the treebank PCFG lexicon, illustrating scarcity of data.

1.1.3 Estimation from unlabeled data

Unsupervised learning of phrase structure grammars has a long history of research, mostly with less-than-positive results. Inside-outside estimation (Lari & Young, 1990) is an instance of the Expectation Maximization (EM) algorithm (Dempster et al., 1977)

and is the mainstay of learning PCFGs from raw (unlabeled) language data. The algorithm takes as input an initial grammar model and a corpus of sentences. It iteratively re-estimates parameters of the model so that the likelihood of the corpus is non-decreasing in each iteration. The algorithm can be used to induce parameter values of the model from scratch or to *re-estimate* parameters of a model obtained by other means. Inside-outside is known to result in sub-optimal grammars – EM is sensitive to initialization parameters and also is known to converge to a local maximum of the objective function rather than search for the global maximum. A wide-coverage PCFG typically has a large number of parameters, with a correspondingly large number of local maxima of the objective function; the algorithm will typically find one of these local solutions. EM is commonly constrained by the use of good initial states, along with heuristic constraints placed on estimated models. In the case of PCFGs, there have been attempts at constraining models by heuristic means such as by stipulating lists of categories that can never be constituents (for example, in Magerman & Marcus (1990)). Another important issue with unsupervised estimation by inside-outside is that there is no clear correspondence between the structures dictated by considerations of numerical maximization performed by the algorithm, and the qualitative judgements of structures desired as the output of parsing. This makes estimated models difficult to evaluate. Most grammars induced by inside-outside have either been toy grammars (for instance, de Marcken (1995)), with the only large-scale grammars being hand-crafted grammars (for instance, Beil et al. (1999). Evaluations have been on test data annotated by the researchers themselves.

In this dissertation, we address these two important issues related to estimation of PCFG's by inside-outside – the issue of imposing constraints on estimated models, and that of objective evaluation of estimated models. We constrain inside-outside in two ways: firstly, the initial model for estimation is obtained by relative frequency esti-

mation from a treebank. This model is an unlexicalized PCFG with complex lexical categories as described before. Since the model is a treebank model, it provides a good initial point for inside-outside estimation. Thus inside-outside is used to *re-estimate* model parameters. Secondly, we make use of the intuition that certain parameters in the treebank model are better estimated than others – these well-estimated parameters can help in the estimation of less-accurate ones. For instance, the syntactic parameters of an unlexicalized PCFG are less scarce than lexical parameters. We therefore re-introduce treebank syntactic parameters in the re-estimated model at each iteration of inside-outside. We also introduce a lexical transformation between iterations that interpolates re-estimated lexical parameters with treebank lexical parameters, preventing re-estimated models from drifting away from the treebank model. In this way, we are able to avoid the use of heuristic constraints; rather, we use the intuition that general syntactic properties and some well-estimated parameters (obtained from a treebank model) can be used for the estimation of more specific lexico-syntactic information.

The use of a treebank grammar as the initial model allows an objective standard of comparison of parses obtained by re-estimated models. Held-out test data from the treebank can be used for the evaluation of re-estimated models, since all re-estimated models have the treebank context-free backbone. This allows standardized evaluations against high quality gold standard data, and comparisons with other results in the field. Re-estimation by maximum-likelihood (via inside-outside) over an unannotated corpus of a treebank model which itself is a maximum-likelihood (relative-frequency) estimate from annotated data allows some examination of questions regarding the relative efficacy of supervised estimation using a treebank and unsupervised inside-outside estimation on a much larger corpus.

Inside-outside estimation is a computationally expensive procedure. The time taken

for one iteration for a grammar with n non-terminals on a sentence of length $|w|$ is proportional to $n^3|w|^3$. An unlexicalized PCFG has a relatively small number of rules, making it possible to use large amounts of unlabeled data to re-estimate models with inside-outside without resort to approximations in either the expectation or the maximization step.

1.2 The structure of the dissertation

The dissertation can be divided into two largely independent parts. The first part is the creation of an accurate unlexicalized PCFG from the Penn Treebank (Chapter 2). The remainder has to do with unsupervised re-estimation of the parameters of this PCFG from unlabeled data.

Chapter 2 is concerned with the building of a Penn Treebank PCFG with linguistic features and representations of empty categories and long-distance dependencies. It includes a description of a Penn Treebank feature-constraint grammar used to build a feature-structure treebank from the Penn Treebank. A PCFG with features incorporated into Penn Treebank category symbols is the created from the feature-structure treebank. The chapter contains evaluations of the treebank PCFG (labeled bracketing and empty category detection) and includes a novel smoothing process used to smooth the treebank model before parsing new data. The process of creation of a linguistically-motivated PCFG from a treebank can be extended to treebanks in other languages. This chapter can be read independently from the rest of the dissertation.

Chapter 3 describes the motivation and theory of a modified inside-outside procedure used to obtain better parameters for a treebank PCFG from unannotated data. It includes a description of the procedure used to build a smoothed treebank model to be

used as the initial model for re-estimation.

Chapter 4 contains a description of the experimental setup for inside-outside re-estimation, descriptions of experiments, and evaluations of results. Evaluations of re-estimated models are on Penn Treebank test data, and focus on learning subcategorization frames for verbs, and labeled bracketing parsing scores.

Chapter 5 contains a summary of the dissertation, and a discussion of ways to continue this research.

Appendix A : List of features and values in the feature-constraint grammar, along with some other details of the feature-constraint grammar.

Appendix B : List of empty categories in the feature-constraint grammar and PCFG models.

Appendix C : PARSEVAL results on some PCFG versions with different incorporated features. PARSEVAL results of PCFG models trained on different amounts of treebank data.

Appendix D : List of subcategorization frames in the treebank PCFG, with example sentences of each frame from the Penn Treebank.

Appendix E : A description of the software components used for conducting the experiments in this dissertation, with the intention of creating a stand-alone package that will allow easy reproduction of the experiment.

CHAPTER 2

TREEBANK FEATURE AUGMENTATION AND PCFG FRAMEWORK

2.1 Introduction

Treebank PCFGs have been the state of the art in parsing technology for the last several years. Treebank PCFGs are those in which the context-free productions of the grammar are obtained from the syntactic annotation on sentences (parse trees) in a treebank. A probabilistic model can also be obtained from the treebank straightforwardly using the relative frequency estimator, simply by reading off counts from treebank trees. Thus, the probability associated with a production of the form $X \rightarrow \alpha$ is : $count(X \rightarrow \alpha)/count(X)$ where $count(X \rightarrow \alpha)$ is the number of times the rule $X \rightarrow \alpha$ occurred in the treebank trees, and $count(X)$ is the count of all rules with left-hand-side X in trees in the treebank. If the probability model of a treebank PCFG is subsequently altered (by using additional data, or data from a different domain for training, etc.), it still remains a treebank PCFG in the sense that if this PCFG is used to parse new data, the parses obtained will have context-free productions of the form present in the treebank.

It is well-known that a vanilla PCFG obtained straight from the trees of a treebank like the Penn Treebank (Marcus et al., 1993) does not perform well, as the context-free assumptions about natural language inherent in such a PCFG are too strong. In order to obtain an accurate PCFG from the Penn Treebank, a useful and well-known strategy is to transform treebank trees in several standard ways – all of them gather either local or long-distance contextual information at the context-free production level in order to weaken context-free assumptions of the treebank grammar. For example, two of the most common transformations are *parent marking* of nodes (Johnson, 1998) and *lexicalization* (Collins (1997), etc.). Parent-marking appends the category of the parent

node in the tree to a category node. Lexicalization marks each category in a tree with its lexical head. A PCFG is then extracted from such a transformed treebank. Full-fledged lexicalization leads to sparse grammars, with various back-off measures used in order to address the problem of data-scarcity.

In this chapter, we describe a framework[1] (Deoskar & Rooth, 2008) for building a high performance unlexicalized treebank PCFG. We use a methodology that involves marking treebank non-terminal nodes with features representing non-local information; the features are both linguistic and tree-geometric in nature. This markup can be thought of as a form of *transformation* of treebank trees before estimation of a PCFG from them (following Johnson (1998)); however our methodology involves the *annotation* of node labels of trees with extra information (in the form of feature-value pairs) rather than changing the structure of the treebank trees. An example of a common structure-transformation is the addition/deletion of NP levels within NP nodes, as is done in the Collins Models (Collins, 1997). In this chapter, we describe the features in the feature-constraint grammar. We evaluate the resulting PCFG using several measures, including the standard PARSEVAL measure. We also provide a limited analysis of the effect of different features on the quality of the resulting PCFG. Our focus is on building a good-quality unlexicalized PCFG to be used as a starting model for the unsupervised estimation experiments that form the bulk of this thesis. In the design of the treebank PCFG, we focus on including features that are lexical in nature, since our experiments with unsupervised estimation are aimed at learning a better probability model for the lexical parameters of the treebank PCFG.

One of the motivations behind the design of this framework is to develop a uniform environment with reusable software components that can be used with treebanks

[1]The treebank PCFG framework described in this chapter is a joint project with Mats Rooth, with some software components written by Lior Privman and Andrew Jonas.

in the Penn Treebank II style in other languages. The existing treebanks of the world's languages have a fairly simple formal vocabulary, consisting of labeled trees with indexed empty categories. The choice of this simple formal vocabulary has certain advantages – for instance, it has spurred research in statistical parsing, and allows simple search methods and search languages. However, for other purposes, like some aspects of linguistic research, lexicographic research and high end parsing, it is a drawback that features like sub-classification of clausal categories and subcategorization features of predicates are not overtly represented in these treebanks. It would also be useful to represent some long-distance dependencies locally on a selecting head, so that statistical models reflects these dependencies. The framework described here is designed in order to facilitate the development of such finer annotations on existing treebanks, creation of feature-grammars and compilation of finer treebank PCFGs for these languages.

2.2 Treebank Feature Augmentation

Our methodology for augmenting the Penn Treebank with features involves constraint-solving with a feature grammar, using the feature-constraint formalism of Yap (Schmid, 2000b,c). A feature-constraint grammar is first constructed from a context-free backbone grammar obtained from Penn Treebank trees (development of the feature-constraint grammar is described in section 2.4). This feature-constraint grammar is used along with a constraint solver Yap (Schmid, 2000b,c) to solve constraints in each treebank tree, effectively resulting in a transformed treebank in which nodes in trees are annotated with with feature structures. Our feature-constraint grammar uses some common constraints that draw upon the long linguistic tradition of feature-based theories like GPSG (Gazdar et al., 1985), HPSG (Pollard & Sag, 1994), LFG (Bresnan, 2001), etc. Probabilistic treebank grammars have been developed in several of these linguistic

formalisms in recent years. For example, O'Donovan et al. (2005) describe a treebank LFG grammar and Miyao et al. (2004) describe a treebank HPSG grammar. While our feature constraint grammar is similar to these with respect to some features, our goal is to use this methodology for treebank transformation and to realize a PCFG in the end. We do obtain a Penn Treebank feature-constraint grammar in the process that may be independently used.

The transformation of treebank trees in this framework can be broken down into the following steps. A detailed description of each step follows in subsequent sections.

1. Preprocessing of treebank trees.
2. Extracting a context-free grammar from the treebank, which serves as the context free backbone to the feature-constraint grammar.
3. Development of a feature-constraint treebank grammar by adding feature constraints to the treebank backbone grammar using perl/lisp scripts, following the Yap feature-grammar formalism (described in Schmid (2000c)).
4. Solving constraints in treebank trees with the feature-constraint grammar; a solution for a tree is represented as a (trivial) shared forest.
5. Building a PCFG from the transformed/annotated treebank.

2.3 Pre-processing of treebank trees

We first obtain a grammar and lexicon from all treebank trees from Sections 0-22 of the Penn Treebank. The parsing research community typically uses these sections from the Penn Treebank as training data, although Section 0 is often used as a development set; we include Section 0 in our training data but hold out every 10th sentence from

Sections 0-22 as a development set[2]. Section 24 is usually used for testing during development and Section 23 for final evaluation. We follow the same practice. The Penn Treebank makes extensive use of empty categories, including A-bar traces, NP traces in control/raising and passive constructions, and traces of various kinds of extrapositions. Our grammar contains all empty category symbols present in the treebank but without the co-indexation information present in the original trees. The Yap feature-constraint formalism that we use includes a provision for empty categories on the right hand side of rules but does not allow for non-terminal nodes properly dominating only empty categories. Since the Penn Treebank contains such tree configurations, the treebank is transformed by substituting unique symbols for such subtrees. Notationally, empty categories in our grammar are flanked by plus signs. For example +T-NP+ designates an A-bar NP trace (for instance a trace of wh-movement), and +E-NP+ designates an NP-trace, or a controlled NP, for instance the trace of a passive. Figure 2.1 shows an example in which the symbol +E-NP+ is substituted for the original tree configuration (NP-SBJ (-NONE- *-1)).

All function tags in the treebank are retained. A section of the bare-bones context-free treebank grammar and associated lexicon extracted from the pre-processed treebank is shown in Figure 2.2. The total number of productions (excluding pre-terminal to terminal productions) is 30903. The lexicon (pre-terminal to terminal productions) has about 44500 entires.

[2]We believe that it is a better strategy to use a sample from all sections of the treebank as a development set, rather than Section 0, whose annotation might not be representative of the annotation in later sections of the treebank.

```
(VP (VBD was)
        (VP (VBN named)
          (S
             (NP-SBJ (-NONE- *-1) )
             (NP-PRD
             (NP (DT a) (JJ nonexecutive) (NN director) )
          )
 )

(VP (VBD was)
        (VP (VBN named)
          (S
             +E-NP+
             (NP-PRD
             (NP (DT a) (JJ nonexecutive) (NN director) )
           )
 )
```

Figure 2.1: An example of converting a Penn Treebank trace (top tree) to our convention (bottom tree).

LHS category	RHS	Terminal	Pre-terminal category
S	NP-SBJ VP -PER-	Pierre	NNP
NP-SBJ	PRP	Vinken	NNP
VP	TO VP	,	-COM-
PP-LOC	IN NP	61	CD
NP	NN	years	NNS
NP	NNS	old	JJ
S	+E-NP+ VP		

Figure 2.2: Barebones treebank grammar rules and lexical entries: Empty categories and function tags are retained in the grammar.

2.4 The Feature-Constraint Grammar

Feature-constraint rules following the formalism of the Yap constraint solver (Schmid, 2000b) are added to the context-free rules obtained from the treebank. In this section, we describe the feature-constraint rules for different context-free rules. The feature-constraint rules are added automatically using scripts which examine patterns in the context-free rule shape. The scripts mostly involve pattern matching using regular expressions and are written in Lisp for VP rules (by Mats Rooth) and in Perl for others (by author). The grammar (excluding lexical, that is, pre-terminal to terminal rules) after addition of all feature constraints amounts to approximately 50000 rules at the current stage of grammar development. In the following subsection (2.4.1), we discuss in detail three important kinds of features and constraints: constraints on auxiliary and other VP rules, addition of valence features to verbs, and tree-geometric features. The subsequent sections document all feature-constraint rules in the grammar.

2.4.1 Features for the auxiliary verb construction

In the Penn Treebank, auxiliary verbs are not distinguished from main verbs. It is commonplace to make this very useful distinction – for example, Collins (1997) and Charniak (1997) rename auxiliary verb tags as AUX. In the Penn Treebank II convention, any local tree with a VP parent and having both VP and verb children is an auxiliary verb construction. For example, a rule such as VP $\rightarrow$ VB VP is an auxiliary rule, with VB being the pre-terminal tag of the auxiliary verb. Hence, we add constraints suitable to auxiliary verb constructions to any rule of this form in the grammar. Feature-constraints in the Yap formalism are enclosed in braces associated with context-free categories (the formalism is described in Schmid (2000b)). A constraint consists of a feature name, an

equal sign and a value, followed by a semi-colon. Feature values in Yap can be constants, variables, lists, and functions. In our grammar, we make use only of constants and variables.

As an example, consider the VP rule shown below (Figure 2.3). The first line shows the bare treebank rule, while the second line shows the corresponding feature-constraint rule with constraints associated with some categories in the context-free rule.

```
VP                                  →  VB ADVP VP

VP { Vform=base; Slash=sl; }        →  'VB { Val=aux; Prep=-; Vsel=vf;
                                             Prtcl=-; Sbj=-;  }
                                       ADVP { }
                                       VP { Slash=sl; Vform=vf; }
```

Figure 2.3: Feature constraints on a VP rule illustrating an auxiliary construction.

The `Vform` feature corresponds closely to the feature by the same name from Gazdar et al. (1985), and takes values like `fin` for finite, `n` for past participle, `g` for present participle and `to` for infinite forms. In this case, it takes the value `base` on the left-hand-side VP. On the right-hand-side (RHS) of the rule, the `Vform` feature on the complement VP (the RHS VP) is equated to a `Vsel` feature (of the same type as `Vform`) on VB via a variable *vf*. A verb phrase (VP) licensed by the above rule will have a `Vform` on the complement VP which correlates with the particular auxiliary verb (VB). For instance, a progressive form of the verb *be* will require the present participle `Vform` value `g`.

The verb is marked as an auxiliary verb with the constraint `Val=aux`, using the `Val` attribute which is also used to describe the valences of non-auxiliary (i.e. main) verbs

(described in the following subsection). The `Prep` (preposition), `Prtcl` (particle) and `Sbj` (subject) features on the verb have a default value in this rule – for VP constructions involving main verbs (i.e., non-auxiliary constructions), `Prep` gets a non-default value if the verb has a PP complement, `Prtcl` gets a non-default value for particle verbs, and `Sbj` characterizes the subject of an S complement. The backquote on VB is a head marking that is required by the Yap formalism, but which is irrelevant to our feature-constraint grammar.

A slash feature `Slash` is used in the standard way (Gazdar et al., 1985) to express *wh* dependencies; the rule matches the `Slash` values of the parent and child VPs using a variable `sl`. The grammar includes slash features on relevant categories (like S, SBAR, VP, etc.) which constrain the distribution of *wh*-traces. Distributions of other empty categories like traces of A-dependencies like passive and raising are constrained by additional features.

2.4.2 Marking valence on verbs

The next example illustrates the encoding of valence (subcategorization) on verbs, and also further illustrates the use of `Slash`, `Vform` and `Vsel` features. Consider the rule in Figure 2.4:

In the VP rule in Figure 2.4, a valence feature `Val` is marked on the verb (pre-terminal category VBN). The valence value `ns` indicates a combination of NP and S complements. The VP rule shows the trace of a passive NP, which in our notation is the empty category +EI-NP+[3], along with a clausal complement S. Note that the trace of the argument NP is indicated in the value of the `Val` feature.

[3]In later versions of our grammar, the empty categories +EI-NP+ (for indexed NP traces) is merged with +E-NP+ (for unindexed NP traces), are merged.

VP → ‘VBN +EI-NP+ S

VP {Vform=n; Slash=sl; Itype=sb} → ‘VBN { Val=ns ; Vsel=vf; Prep=-;
Prtcl=-; Sbj=sb; }
+EI-NP+
S { Sbj=sb; Slash=sl; Vform=vf; }

Figure 2.4: VP rule illustrating valence feature `Val` on verb with +E-NP+ S complement. +E-NP+ is the trace of an unindexed NP in our grammar.

We use a vocabulary of 31 basic valence values. A full key to all valence feature values and their associated structures is shown later in Table 2.1.

The subject of the clausal complement (indicated by `Sbj`) is equated using the variable `sb` to features on both the parent VP and the child VBN, thus propagating it to the VP level and marking it on the pre-terminal category of the verb. The same is the case with `Vform` on the complement S. The value of the `Slash` feature is matched between the parent and the S child, indicated by the use of the variable `Slash=sl` on the RHS and LHS of the above rule. The features `Prep` and `Prtcl` stand for *preposition* and *particle* and get values corresponding to the lexicalized preposition or particle in the case of the valence including a PP or a PRT complement. In the above rule, they have the default value of “-”.

The equation `Vform=n` on the parent identifies the passive verb phrase. Note that the passive verb phrase is distinguished from the “have” VP. That is, the VPs in sentences like “The book [$_{VP}$is written by John]” and “John [$_{VP}$has written the book]” are distinguished. In the “have” case, the verbal category dominating a form of “have” is marked with `Vsel=h` and the VP is marked with `Vform=h`. In general, dependencies

such as passive and raising are constrained with local features such as `Vform` and `Vsel`, reserving the slash feature for A-bar dependencies.

The next example (in Figure 2.5) also illustrates the features involved in a passive verb, in this case with an NP complement (i.e. a transitive verb). The value of the `Vform` feature on the VP is `n` to indicate a participial verb VBN on the RHS. Again, the trace of the passive is the empty category +EI-NP+. The value of the `Val` feature is `n` (standing for a noun phrase complement, with the noun phrase being empty +EI-NP+ in this case). Note that the PP on the RHS is not included in the valence of the verb. Only PP-CLRs (which indicate complement prepositional phrases in Penn Treebank annotation) would be included in the valence of the verb. In this case, unlike the previous example, the `Slash` feature is not involved (and gets the default value of "-"). The features `Sbj` and `Vsel` also have a default value (since there is no clausal complement), as also `Prep` and `Prtcl`.

The other features appearing in Figures 2.4 and 2.5 are explained in subsequent sections.

VP	→	'VBN +EI-NP+ PP
VP {Vform=n; Slash=-; Itype=-;}	→	'VBN { Val=np; Vsel=-; Prep=-; Prtcl=-; Sbj=-; } +EI-NP+ PP {parent=vp;}

Figure 2.5: VP rule illustrating valence feature `Val` with value `np` on verb.

Table 2.1: Values of the `Val` feature marked on verbs. X-PRD refers to categories with a predicative functional tag like NP-PRD, ADJP-PRD, PP-PRD, etc.

aux	VP
b	SBAR
bn	NP SBAR
bnp	NP PP-CLR SBAR
bp	PP-CLR SBAR
br	PRT SBAR
d	PP-DIR
de	NP-EXT PP-DIR
dn	NP PP-DIR
dnr	NP PRT PP-DIR
dr	PRT PP-DIR
e	NP-EXT, PRT NP-EXT
en	NP NP-EXT
ep	NP-EXT PP-CLR
m	NP NP
mp	NP PP-CLR NP
n	NP
np	NP PP-CLR
npr	PRT NP PP-CLR
nr	PRT NP
ns	NP S
nt	X-PRD NP , NP X-PRD
p	PP-CLR
pr	PRT PP-CLR
ps	PP-CLR S
r	PRT
rs	PRT S
rt	PRT X-PRD, X-PRD PRT
s	S
t	X-PRD
z	intransitive

Projecting structural information onto lexical items

The above rules in Figures 2.4 and 2.5 illustrate our general strategy of projecting information about the tree shape selected by a lexical item on to the lexical category. For instance, in Figure 2.4, the past participle (VBN) verb will get marked with the valence value `ns`, with `Sbj` and `Vsel` values copied from the S complement, and default values for the features `Prep` and `Prtcl`. This sequence of features indicates the structure of the complement of the verb. Note that even the structure embedded inside the S complement gets marked onto the verbal pre-terminal tag.

Thus, the basic valence of a verb, indicated by `Val`, is sub-classified due to the presence of additional features on the verb. The `Vsel` feature marks the `Vform` of a complement S, or for auxiliary verbs, the complement VP. This distinguishes, for instance, control verbs like *try* which select an S with `Vform=to`. The `Sbj` feature marks whether the complement, if it is an S, has an empty subject. For example, a control use of the verb *try* like in the fragment *tried to leave* has `Sbj=e`, marking a null +EI-NP+ subject. The verb *considered* in the sentence *they are officially considered strategic* gets pre-terminal values of `Val=s`, `Sbj=ei`, and `Vsel=sc`. These values indicates a clausal complement (`s`) which has an empty subject (`ei`) since the sentence is passive and is of the type *small clause* (`sc`). There are 81 realized combinations of values for `Val`, `Vsel`, and `Sbj`, providing a moderately fine-grained classification of valences. The features `Prtcl` and `Prep` further sub-classify verbs with particle and prepositional complements, by indicating the particular choice of particle or preposition. In a method described in the following sections, the effect will be to construct a lexicon with fairly specific information about the tree shapes associated with lexical items, using information implicit in the treebank structures.

2.4.3 Tree-geometric features

We also use features which are tree-geometric rather that linguistic in nature, in the style of Johnson (1998) and Klein & Manning (2003). These are features that mark contextual information onto a node and are relevant to producing a good PCFG model. An example is the `Vdom` feature marked on ADJP-PRD (predicative adjective phrase) in the rule below. The value `vd` for this feature has the interpretation that the bearer of the feature directly or indirectly dominates VP. Similarly, the `parent` attribute on PP is a tree-geometric contextual feature marking the upward context (i.e., the parent node).

ADJP-PRD	→	ADJP PP S
ADJP-PRD {vdom=vd;}	→	‘ADJP { } PP { parent=adjp;} S { };

Figure 2.6: Rule illustrating a tree-geometric feature

2.4.4 Some comments on the design of the feature-constraint grammar

Since the feature-constraint grammar is based on a treebank backbone, it has a large number of rules. The Yap formalism does not allow for factoring of constraints in an inheritance hierarchy (like in HPSG (Pollard & Sag, 1994), for example); hence the rules have redundant patterns of feature constraints. This has some disadvantages in grammar development, since there is no concise localization of a given constraint. At the same time, the grammar development environment proves to be a comfortable one, because the treebank nearly eliminates the issue of ambiguity. This allows the computational linguist to concentrate on correct analyses while developing the feature-constraint

grammar. We envision this setup as a simple and easily deployable platform for augmenting existing treebanks with features and creation of feature-constraint grammars and PCFGs.

The design for the grammar is motivated to a large extent by the PCFG compilation application, described in §2.7. In particular, issues of sparse distributions arising from splitting treebank rules into multiple finer rules (and the multiplicative effect of features on more than one RHS categories) are an important consideration. This has consequences for the complexity of the feature analyses – notably, only atomic-valued features are employed in the grammar. It is clear that a grammar at this limited level of complexity misses linguistically real phenomena, and thus should be regarded as an approximation. Our aim is to strike a balance between linguistic sophistication, statistical considerations arising from the limited size of treebank data, and computational and mathematical simplicity and tractability.

2.5 Description of all features in the feature-constraint grammar

This section lists all implemented features in the feature constraint grammar, along with their possible values, and a brief description of each.

2.5.1 General structure-related features

Parent Feature

Johnson (1998) obtained a sharp improvement (approximately 7% in precision and 10% in recall) by labeling all categories with their parent categories. His analysis also showed

that distinguishing S nodes with a Root or non-Root parent was highly beneficial, as was marking NP nodes with their parent. Following Johnson (1998), we add a parent feature to select categories S, VP, ADJP-X, ADVP-X, WHADJP, WHADVP, NP, WHNP and QP. This feature takes values out of the set {s, vp, adjp, advp, np, qp, -} depending on the parent of these categories. The feature values are self-explanatory.

Categories S, SQ, SINV and SBAR have a `parent` feature with the value `root` if they are daughters of ROOT. Distinguishing root and non-root S categories in this way predictably improved f-score of the PCFG substantially. One of the benefits of this feature is that it helps to attach the punctuation category -PER- to the top most S category; -PER- is always the daughter of the topmost S node and not of a lower S node in Penn Treebank II annotation.

VDom (Verb Dominating)

Following Schmid (2006), nodes of a non-verbal category (i.e. all categories except VP) dominating VP, SINV, S, SQ, SBAR, SBARQ nodes are distinguished from those that do not dominate these categories, using a feature called `vdom`. This feature takes two values: `vdom=vd` when non-verbal nodes dominate VP, SINV, S, SQ, SBAR, SBARQ, and `vdom=nvd` when they do not.

2.5.2 Features on nominal categories

Distinguishing base and non-base NPs

We distinguish base NPs (those dominating a nominal category) from non-base NPs with a feature called `nptype`, following Collins (1997), Klein & Manning (2003) and

Schmid (2006). All NPs which dominate a category of NN, NNS, NNP, NNPS, DT, CD, JJ, JJR, JJS, PRP, RB, EX (that is, those NP rules which have these categories on their RHS) are marked with the value `nptype=base`, while the rest are marked with `nptype=-`. Collins (1997), in addition to marking base NPs, adds extra structure to some NPs to reduce the perplexity of the NP portion of the model (Bikel, 2005). When an NP is relabeled as a base NP in Collins (1997), a normal NP node is inserted as a parent in some cases (when the parent of the base NP is not an NP, or is an NP but is in a co-ordinated phrase). Also, clause nodes that are sisters to nominal nodes are moved higher and attached as sisters to a base NP node. We do not do any transformation of NP nodes, by either adding or deleting structure.

Genitive NPs

All NP rules with the category POS on the RHS are marked with the feature `pos` (with a value `y`) to distinguish genitive NPs.

Percent feature

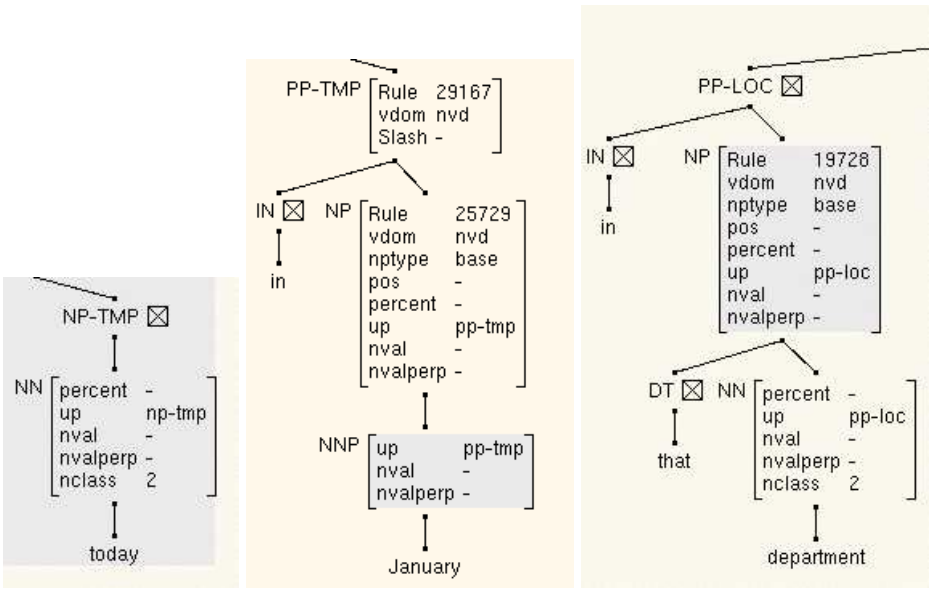
All NPs that dominate an NN that dominates the percent sign are marked with a feature `percent=perc`. In all other cases, the value of the percent feature is the default "-". NPs get this feature set by propagation from the daughter NN category, which in turns gets it from the lexicon. The lexical entry for "%" in the lexicon has the feature `percent=perc;`, while all other words in the lexicon have the feature `percent=-`. Thus, "%" is effectively lexicalized.

Up feature

In order to distinguish temporal and locative nouns (those which occur under noun and prepositional phrase categories like NP-TMP, PP-TMP or PP-LOC) from other nouns, all pre-terminal nominal categories (NN, NNS, NNP and NNPS) that are under NP-TMP, PP-TMP or PP-LOC nodes are marked with a feature `up`. The `up` feature takes values from the set `up=` {`np-tmp, pp-tmp, pp-loc, -`}. This feature thus marks nouns in the lexicon as temporal or locative, based on their occurrence as daughters of temporal or locative NPs. This feature is an example of information being passed down into the lexicon, since the feature is marked on pre-terminal noun categories. Temporal and locative nouns have attachment patterns that may be different from other nouns, and might help with distinguishing complement NPs and PPs from adjuncts. For example, a temporal noun may be more likely as an adjunct than an argument. Example of the `up` feature being marked on nouns that are daughters of NP-TMP, PP-TMP and PP-LOC are shown in Figure 2.7

Noun Valence feature

Another lexical feature on nominal categories is the noun valence feature `nval` , which indicates the category of the complement of the noun. Currently, this feature takes four values: `s` for an S complement, `p` for a PP complement, `sbar` for an SBAR complement, and a default value of "`-`". In Penn Treebank II annotation, PP complements of nouns are not attached as sisters of the nouns themselves, but are attached as sisters of the parent NP (all postmodifies of nouns are Chomsky-adjoined in Penn Treebank II annotation, except for clausal (S and SBAR) complements). The `nval` feature in the case of PP complements thus needs to be propagated down to the nominal category through the parent NP node. Examples of the required constraints on the two rules are shown below



(a) NP-TMP (b) PP-TMP (c) PP-LOC

Figure 2.7: The `up` feature on nouns.

in Figure 2.8. The first rule adds the feature `nval=p` to the NP category on the RHS to indicate the complement PP (the PP is the complement of the second NP on the RHS). The specific preposition marked on the PP category is equated to a feature called `nvalperp` on the RHS NP using a variable called *pp*. The second rule equates the values of the `nval` and `nvalperp` features from the LHS NP-SBJ to the NNS category on the RHS via the two variables *nv* and *pp*.

S and SBAR complements on the other hand, are marked as sisters of the nouns, and the `nval` feature in these rules only needs to be equated between the noun category and the S or SBAR category. The three noun valences and the structures associated with them as shown in Figure 2.9.

NP-SBJ {vdom=nvd;nptype=-; pos=-; percent=-; } →

‘NP { up=-; } CC {}

NP { up=-; nval=p; nvalperp=pp;}

PP { parent=np; Perp=pp; } ;

NP-SBJ {vdom=nvd;nptype=base; pos=-; percent=-; nval=nv;nvalperp=pp;} →

‘ADJP { } JJ { }

NNS { up=-; nval=nv; nvalperp=pp; } ;

Figure 2.8: Two rules involved in marking a PP complement on a noun

List of all features and their types on Nominal categories

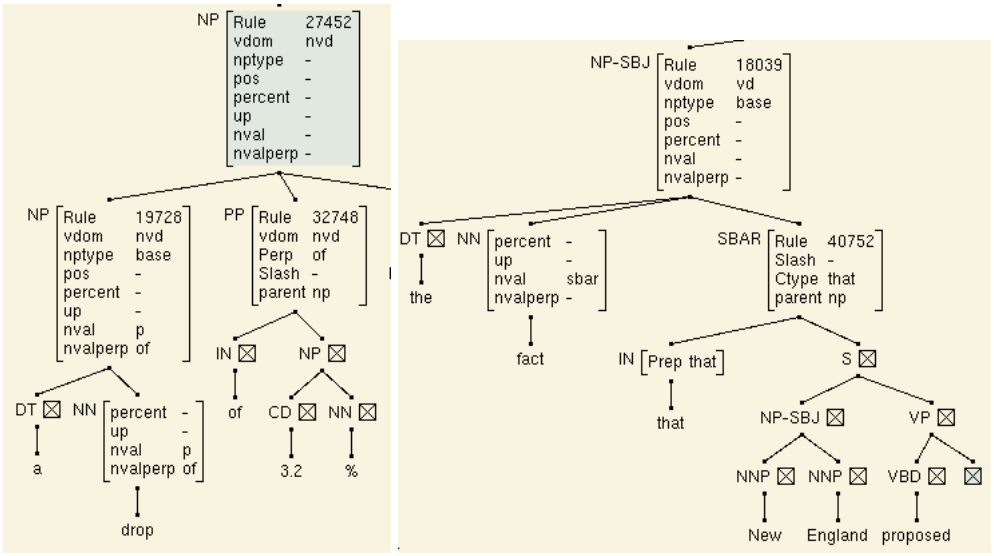
A list of all features and their types on nominal categories is shown in Figure 2.10.

2.5.3 Clausal categories

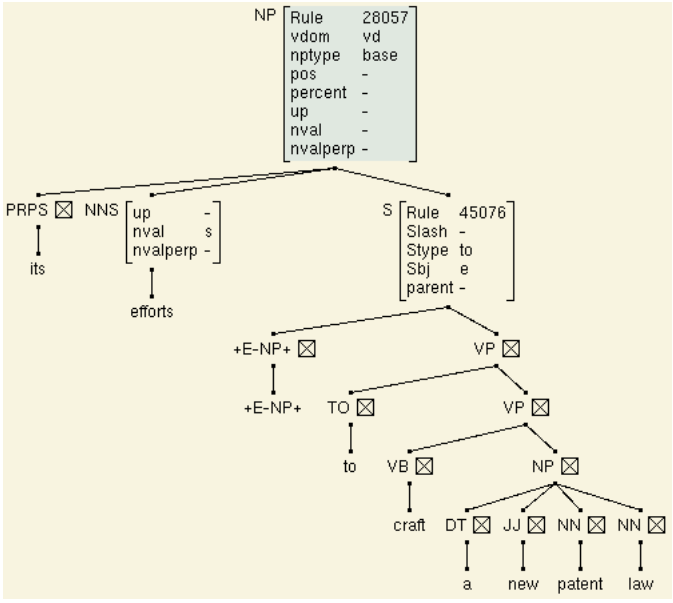
S Categories

The category S (with or without a function tag) has four features: `Slash, Stype, Sbj` and `Parent`. The `Stype` feature indicates the type of clause and has the same type as the `Vform feature` on VPs. The `Stype` feature is marked on the categories S, S-TPC, S-CLR, S-ADV and S-NOM. The feature `Sbj` refers to the subject of the clause and can take a value out of the set {e, t, -}). The value `e` refers to subjects that are the empty categories +E-NP+ or +EI-NP+[4]. The value `t` indicates the empty subject category +T-NP+, or other A-bar traces like +T-S+. The value “-” refers to a non-empty subject.

[4]In later versions of our grammar, these two empty categories are merged.



(a) Prepositional valence (b) SBAR valence



(c) S valence

Figure 2.9: Different values of the valence feature on nouns.

NP	VDOM vdom; NPTYPE nptype; POSS pos; PERCENT percent ; UP up; NVAL nval; PERP nvalperp;
NN	PERCENT percent; UP up; NVAL nval; PERP nvalperp;
NNP, NNS, NNPS	UP up; NVAL nval; PERP nvalperp;

Figure 2.10: List of all features on nominal categories

`Stype=inf, Sbj=e`	Infinitive Clause	S – > X +E-NP+ X VP X
`Stype=n, Sbj=t`	Relative Clause	S – > X +T-NP+ X VP X
`Stype=fin, Sbj=-`	Finite clause,	S – > X X NP-SBJ X X VP X X
`Stype=to, Sbj=-`	Active ECM	S – > NP-SBJ X VP X
`Stype=sc, Sbj={e,t,-}`	Small Clause	S – > NP-SBJ X X-PRD X X
`Stype=scclr, Sbj=e`	Closely related	S-CLR – > +E-NP+ X-PRD X X
	small clauses	(e.g. *Stocks closed higher.*)

Figure 2.11: List of all features on S categories

The value of `Stype` and the associated form of rule is shown below in Figure 2.11 (X is a filler for any sequence of adjunct categories).

Sbar Categories

Three features are marked on SBAR categories : `slash`, `Ctype` and `parent`. The `Ctype` feature gets values based on the complementizer under IN under SBAR [5].

- If the preposition[6] (that is, the value of the feature `Prep` on the IN category under

[5]The design of these features was reverse-engineered to some extent by examining an older and undocumented version of `Sbar` feature-constraint rules by Yuping Zhou.

[6]This is actually the complementizer, or what is called the subordinating conjunction in traditional

SBAR) has the values than, that or whether, then propagate the value to the feature Ctype.

- If Prep ∈ {as, for, since}, Ctype =ambgs
- If Prep ∈ {after, although, because, before, if, in, like, to, until, while, with,-} , then Ctype = dSVP .
- For rules with SBAR categories on the LHS and RHS, use a variable Slash= sl on SBAR categories to propagate the value of the Slash feature from the RHS SBAR to the LHS SBAR.
- For rules with two SBARs on the RHS (conjunctions), leave Ctype unconstrained. Forcing it to the default value leads to parse failures, since the two might have different values.

2.5.4 Prepositional categories

There are four features on PP: vdom, Perp, Slash and parent. The category IN has the feature Prep, which lexicalises certain prepositions by propagating them up from the lexicon and marking them on IN. The values that can be taken by the Prep feature are from the set {about, after, against, among, although, as, at, because, before, between, by, for, from, if, in, into, like, of, on, over, since, than, that, through, to, under, until, whether, while, with, -, _ } .

grammars. We call it a preposition because the POS tag in the Penn Treebank of these complementizers is IN (the same as for prepositions). We use the same feature Prep to lexicalise these complementisers as we do for prepositions.

2.5.5 Dollar

The category QP is marked with a feature `u=dol` if it dominates categories like -DOL- or -HSH-. The QP category that dominates a -DOL- or -HSH- category is usually the sister of an empty category +U+, according to treebank annotation, as shown in Figure 2.12. In addition, if the RHS of a QP rule contains a CD NN combination, add `u=dol` feature to QP. These rules capture forms like “4 %” and “4% to 5%”, etc. The `u=dol` feature is also added to NP (and a few ADJP) rules which contain +U+ and QP on the RHS, but not DOL or HSH.

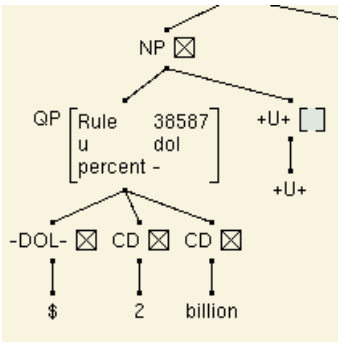


Figure 2.12: Feature on QP

2.5.6 Determiners

The determiners *a, an, the, this, that, those*, and *these* are lexicalized. The category DT has a feature `dtype` whose value is obtained from the lexicon. The value is one of `a`, `an`, `the`, `this`, `that`, `those`, or `these` for these determiners, and is the default “-” for other determiners.

2.5.7 Punctuation

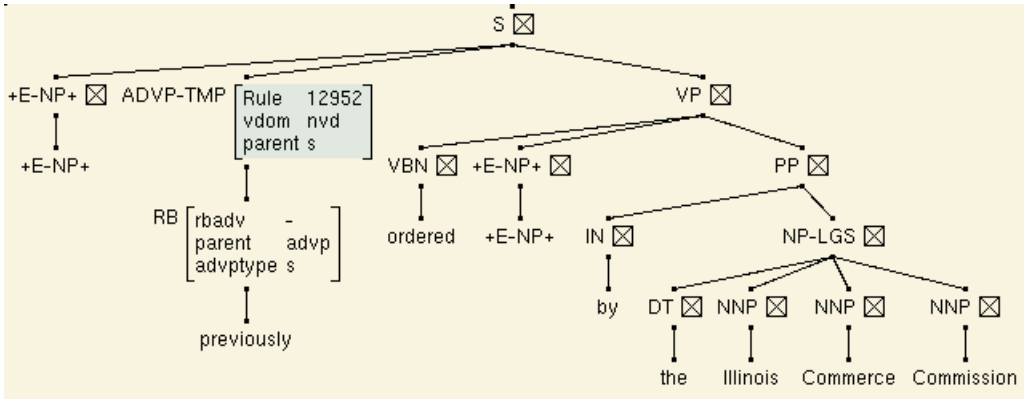
Following Schmid (2006), we split the category COL into different categories depending on what punctuation it dominates, by adding the `col` feature, which takes one of three values : `stop, ques`, or `excl`.

2.5.8 Adverbs

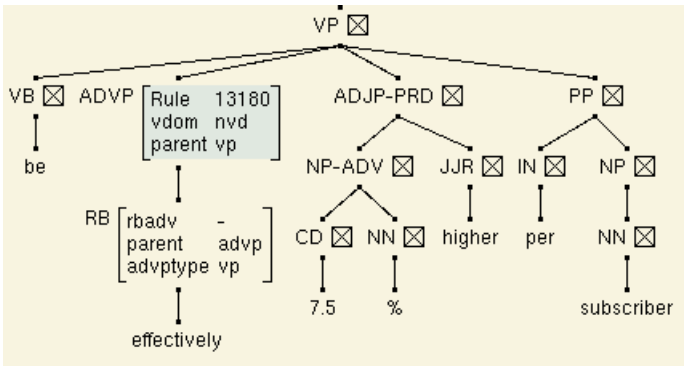
Following Schmid (2006), the pre-terminal category RB is lexicalized if the terminal adverb is *as*, *so*, *about*, or *not* by adding the feature `rbtype`, whose values are obtained from the lexicon. In addition, ADVP categories are marked with a parent feature that indicates the parent node of the ADVP. If the parent is a clausal category (S), a VP or an NP, the value of the `parent` feature is passed down to the pre-terminal RB category as the feature `advptype` to mark the adverb as a clausal, verbal or nominal adverb. Figure 2.13 illustrates the structure associated with these values. This feature is an example of a lexical feature which gets its value from structural information passed down the tree.

2.5.9 Verbal categories

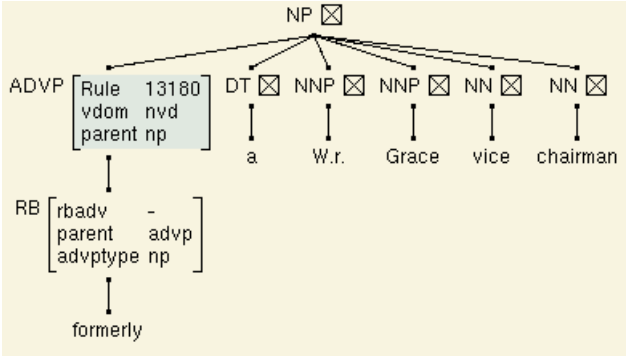
Rules for auxiliary VPs and for adding valence features to verbal categories in VP rules were described in sections 2.4.1 and 2.4.2. The list of all features on VP categories is shown in Figure 2.14. Grammar development for VP rules was done by Mats Rooth. Refer to Deoskar & Rooth (2008).



(a) Sentential adverb



(b) Verbal adverb



(c) Nominal adverb

Figure 2.13: Different values of the `advptype` feature on category RB.

Vform	The feature Vform is marked on the VP category. Values of Vform are from the set: {fin, base, n, h, g, to, sc, scclr, -}. Vform can also take the values {wnn, wnns, wnnp, wjj, wpos}, used for tag errors in VP rules.
Vsel	Vsel is a feature of of the same type as the vform feature. It is marked on verbal categories on the RHS of VP rules. Values that can be taken by the Vsel feature are the same as the Vform feature.
Val	Valence features, values of Val are listed in Table 2.1.
Prt	Indicates the particle in a particle complement.
Perp	Indicates the preposition in a prepositional complement.
Sbj	Subject of an S complement.

Figure 2.14: List of all features on verbal categories

2.5.10 Slash propagation mechanism

Slash propagation for categories except VP is used for A-bar dependencies and is implemented by a Slash feature. The constraints related to the Slash feature are added by scripts, based on the presence of empty categories in the RHS of a rule, or propagated from the RHS of a rule to the LHS when appropriate. The categories that are marked with a Slash feature are ADJP, PP, S, SBAR, SBARQ, SINV, SQ, VP and their functional categories. The values that the Slash feature can take are: n, adv, adj, p, v, s, sbar, ssbar, sbarq, nv, sinvsbar, frag, sq, advpvp, sinv, ucp, sbarsbar, -.

A relative clause, an example of an A-bar dependency, is illustrative of the working

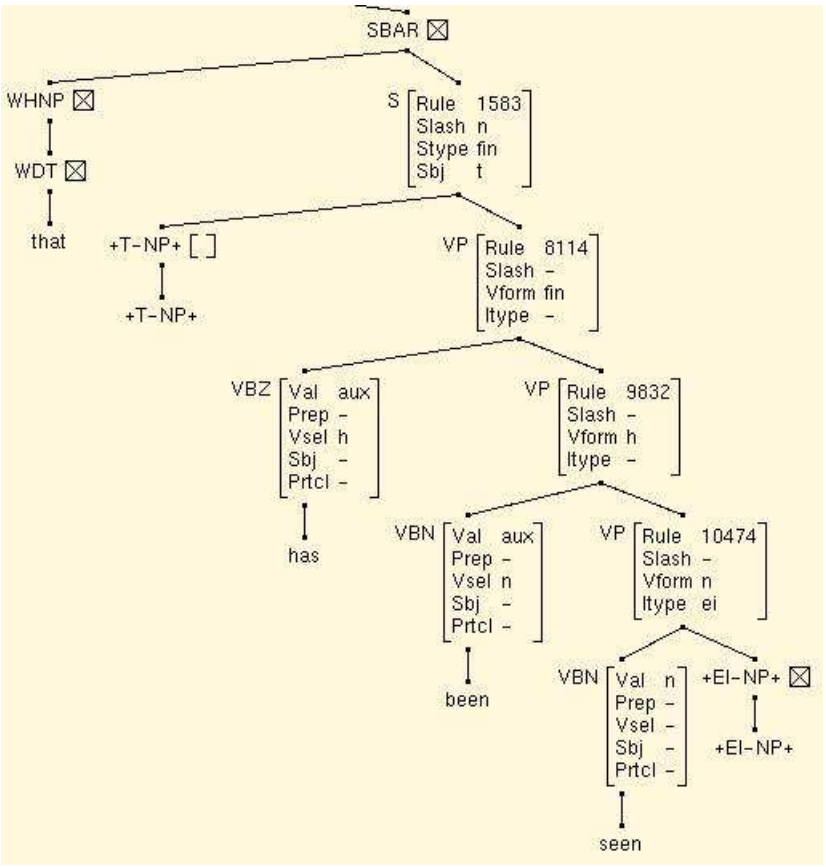


Figure 2.15: A relative clause in the transformed treebank: Empty categories are flanked by plus signs.

of the `Slash` feature and is shown in Figure 2.15. The `Slash` feature is not involved in the rules in the lower parts of the tree, where the dependencies are A-dependencies (example, trace of a passive). The trace of the subject of the relative clause is +T-NP+ and is indicated by the value `Slash=n` on the S node. The `Slash` feature will get cancelled (`Slash=-`) higher in the tree, when the noun that the relative clause is modifying is encountered.

2.6 Transforming the Penn Treebank by solving constraints

All trees in the Penn Treebank (Sections 0-23) are converted into feature structure trees, giving us a feature-structure treebank database. The following steps are involved.

Building the feature-constraint grammar

First, context-free rules and lexical entries annotated with feature-constraint rules as described above are compiled using the YAP compiler (Schmid, 2000b) into a set of three files (.gram, .lex, .fs). The .gram file is the context-free backbone of the feature-constraint grammar, while the .lex file contains simply a listing of lexical items. The feature specifications are in binary format in the .fs file.

Transforming the treebank

The methodology for transforming the treebank trees into trees annotated with feature-structures can be broken into two stages. In the first stage, for each tree in the treebank, a trivial shared forest data structure is constructed, represented as files with a .cpf extension. This forest represents the single tree licensed by the context-free backbone grammar whose yield is the sentence. In the second stage, constraints are solved in the shared forest, using the Yap constraint solver (Schmid, 2000b) along with the compiled Yap feature-constraint grammar described above. This stage adds features and may split a tree into several solutions if feature-constraint rules are ambiguous. The output of this stage is stored in files with the extension .fpf.

Figures 2.15 and 2.16 show parts of sample trees in the transformed treebank. The

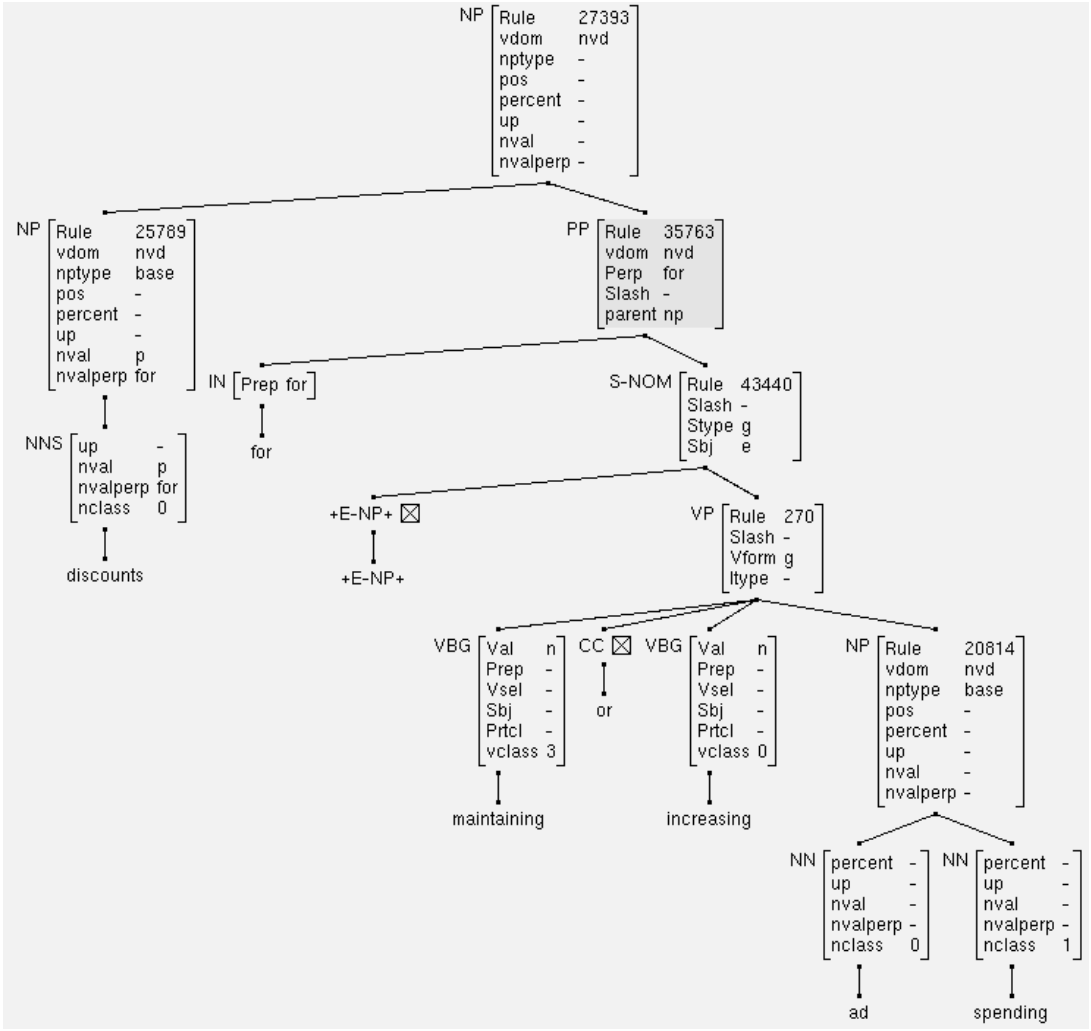


Figure 2.16: Prepositional complements of nouns are marked on the noun *discounts* (`nval=p`) along with the specific preposition (`nvalperp=for`). The tree also shows the transitive valence `Val = n` marked on the verbs *maintaining* and *increasing*. The features `nclass` and `vclass` are described later in §2.11

node labels and tree shape are as in the original treebank, except for simple transformations related to empty categories (described in section 2.3). Each node is annotated with a feature-structure associated with that category label. The feature values make explicit certain information that is implicit in the treebank. Figure 2.15 shows a relative clause, containing VP nodes with `Vform` features taking different values. Notice that features are marked on lexical items – for instance the valence feature on verbs. The auxiliary verbs in Figure 2.15 are marked with `Val=aux`, while the main verb *seen* has a `Val` feature whose values depends on its complement (a transitive verb with `Val=n` in this case). Figure 2.16 shows an NP with a prepositional complement marked on the head noun *discounts* (`nval=p; nvalperp=for`). It also shows the transitive valence `Val = n` marked on the verbs *maintaining* and *increasing*.

2.7 PCFG Compilation

PCFGs are often created by incorporating features into context-free grammar symbols (for example, Klein & Manning (2003)). These symbols along with the incorporated features then form the new context-free symbols of the PCFG. We implemented a method in which for each treebank non-terminal symbol, a list of attributes (features) to be incorporated in it is stipulated. For instance, it may be stipulated that VP incorporates the attributes `Vform` and `Slash`, and that verbs incorporate valence `Val` and `Vform`. This set of attributes (features) stipulated for each non-terminal category is referred to as the "incorporation sequence" for the non-terminal in the PCFG.

A program reads the shared forest structures produced by constraint solving, and collects frequencies of occurrences of local tree configurations, including context free symbols and values of features to be incorporated into the context-free symbols. In

!	-PER-.excl 56.0
#	-HSH- 131.0
$	-DOL- 6773.0
%	NN.perc.p.- 1286.0 NN.perc.-.pp-tmp 1.0 NN.perc.-.- 3270.0 NN.perc.p.pp-loc 5.0 JJ 8.0 NN.perc.-.np-tmp 5.0 NN.perc.-.pp-loc 14.0
&	SYM 1.0 CC 937.0
’	-RDQ- 124.0 POS 567.0 -COL- 3.0
”	-RDQ- 6335.0
’30s	NNS.-.pp-tmp 1.0 CD 1.0
’40s	CD 1.0 NNS.-.- 1.0
’60s	NNS.-.- 3.0
abandon	VB.np.-.-for 1.0 VB.n.-.-.- 11.0
abandoned	VBN.aux.e.fin.- 6.0 VBN.n.-.-.- 9.0 VBD.n.-.-.- 13.0 JJ 1.0
abandoning	VBG.n.-.-.- 5.0
abandons	VBZ.n.-.-.- 2.0
abate	VB.z.-.-.- 2.0
abatement	NN.-.-.- 3.0
abates	VBZ.z.-.-.- 1.0
abating	VBG.z.-.-.- 1.0 VBG.aux.e.fin.- 1.0
Abb	NNP.-.- 5.0
Abbe	NNP.-.- 1.0
Abbie	NNP.-.- 8.0
Abbott	NNP.-.- 3.0
tried	VBN.n.-.-.- 5.0 VBD.z.-.-.- 1.0 VBN.aux.e.fin.- 1.0 VBD.n.-.-.- 1.0 VBN.s.e.to.- 11.0 VBD.s.e.g.- 1.0 VBN.z.-.-.- 1.0 VBD.s.e.to.- 32.0
tries	VBZ.s.e.to.- 14.0 NNS.-.pp-tmp 1.0
trifle	RB.-.np.- 1.0
trigger	NN.-.p.np-tmp 0.333 VB.n.-.-.- 4.0 NN.-.p.pp-tmp 0.333 NN.-.p.pp-loc 0.333 VBP.n.-.-.- 1.0 NN.- .-.- 4.0
triggered	VBN.n.-.-.- 15.0 VBD.n.-.-.- 17.0 JJ 1.0

Figure 2.17: Examples of entries in the PCFG lexicon: each word is listed along with the pre-terminal symbols (treebank POS-tag followed by the sequence of incorporated features) with which it has occurred in the transformed treebank and their frequency.

29092	ROOT	→	S.fin.-.-.root
14134	S.fin.-.-.-	→	NP-SBJ.nvd.base.-.-.- VP.fin.-.-
13057	NP-SBJ.nvd.base.-.-.-	→	PRP
13050	PP.nvd.of.np	→	IN.of NP.nvd.base.-.-.-.-
11226	S.fin.-.-.root	→	NP-SBJ.nvd.base.-.-.- VP.fin.-.- -PER-.stop
10760	VP.to.-.-	→	TO VP.base.-.-
10267	NP.nvd.-.-.-.-.-	→	NP.nvd.base.-.-.p.- PP.nvd.of.np
7099	S.to.e.-.-	→	+E-NP+ VP.to.-.-
6474	NP.nvd.base.-.-.-.-	→	NN.-.-.-
6374	VP.fin.-.-	→	MD VP.base.-.-
5588	NP.nvd.base.-.-.-.-	→	NNS.-.-
5583	PP-LOC.nvd.in	→	IN.in NP.nvd.base.-.-.-.pp-loc
5494	NP.nvd.base.-.-.-.-	→	DT.the NN.-.-.-
4782	S.fin.t.n.-	→	+T-NP+ VP.fin.-.-
4774	SBAR.nulcmp.-.vp	→	+C+ S.fin.-.-.-
4698	SBAR.whn.n.np	→	WHNP S.fin.t.n.-
4653	NP-SBJ.nvd.base.-.-.-	→	DT.the NN.-.-.-
4289	PP.nvd.of.np	→	IN.of NP.nvd.-.-.-.-.-
4131	NP-SBJ.nvd.base.-.-.-	→	NNP.-.- NNP.-.-
3860	QP.dol.-	→	-DOL- CD CD
3834	WHNP	→	WDT
3716	NP.nvd.base.-.-.-.-	→	JJ NNS.-.-
3641	S.fin.-.-.root	→	NP-SBJ.nvd.-.-.-.- VP.fin.-.- -PER-.stop
3471	NP.nvd.base.-.-.p.-	→	DT.the NN.-.p.-
3451	NP.nvd.base.-.-.p.-	→	NNP.-.- NNP.p.-
3436	VP.base.-.-	→	VB.n.-.-.- NP.nvd.base.-.-.-.-

Figure 2.18: The most frequent syntactic rules and their frequencies in the treebank PCFG.

cases where constraint solving introduced ambiguity, frequencies are split by a non-probabilistic version of the inside-outside algorithm, the ratio algorithm. The result is a rule frequency table and frequency lexicon which can be used by a probabilistic context-free parser. The process is implemented as follows:

- First, each .fpf file is converted using the program *yappffun* (Privman, 2003) into a corresponding .flow file, which represents the treebank tree obtained by constraint solving with stipulated features incorporated into treebank categories. In the case

Lexical entry	abandon	VB.np.-.-.for 1.0		VB.n.-.-.- 11.0
Pre-terminal rules	1.0	VB.np.-.-.for	→	abandon
	11.0	VB.n.-.-.-	→	abandon

Figure 2.19: Translation from a lexical entry for a word in our PCFG to pre-terminal rules associated with that word.

of ambiguity, all features are included as a subjunction on a category.

- The .flow files to be used for training the PCFG are moved to a single directory, and the program *flexbuild* (author: Andrew Jonas) is used to compile the PCFG from all flow files in the directory.

Examples of the resulting grammar and lexicon are shown in Figure 2.18 and Figure 2.17 respectively. Both the lexicon and grammar are represented as a frequency table from which a relative-frequency probabilistic model can be constructed. In the case of the lexicon, each word is listed along with the pre-terminal symbol with which it has occurred in the transformed treebank and the frequency of occurrence of the pair. This pre-terminal symbol now consists of the original treebank part-of-speech (POS) tag followed by the sequence of incorporated features. For instance, PCFG pre-terminal rules constructed from a lexical entry such as the one for the verb *abandon* are shown below in Figure 2.19.

2.8 Smoothing of Grammar Productions

Context-free productions extracted from the Penn Treebank contain a large number of low-frequency rules having long right-hand-sides, due to flat structures in the Penn Treebank trees. In order to reduce the associated sparsity, these long rules are generally bro-

ken down into binary rules (for example, Klein & Manning (2003)). In our treebank PCFG, we do not implement a smoothing scheme for the grammar productions of the treebank PCFG. Since the grammar productions are not lexicalized, this is not such a severe problem. However, a horizontal markovization of long right hand sides of rules is expected to improve the labeled bracketing f-score of the PCFG. For example, both Klein & Manning (2003) and Schmid (2006) obtain an increase of almost 1% in labeled bracketing f-score by breaking down long right-hand-side rules into binary rules.

2.9 Smoothing of PCFG Lexicon

Our PCFG contains very fine lexical categories, due to incorporation of features such as valence in the POS tags of lexical items. This implies that the frequency models associated with lexical categories are extremely sparse. In order to parse new data, we need a smoothing scheme that will take frequency from seen items and assign it to unseen items. We implement a smoothing scheme in which we depend upon a standard POS tagger to first tag the data to be parsed (i.e., the test data) with standard Penn Treebank style POS tags. Once the test data is POS tagged, we smooth the PCFG lexicon in a way so as to address two issues:

- Word and POS tag combinations that are present in the test data but not in the PCFG lexicon must be added to the lexicon, with some probability mass allocated to the new entries. Since the word and POS tag combination is unseen, all incorporation sequences associated with that tag are also unseen for that word. Hence, the smoothed lexical entry for each such word must be contain all possible incorporation sequences for that tag.
- In the case of word and tag combinations in the test data that do exist in the PCFG

lexicon (i.e., seen combinations of word and POS tag), the test data might contain configurations for which tag-incorporation combinations (for a given word) have not previously occurred in the treebank training data. It is thus necessary to add all possible tag-incorporation combinations to the lexical entry for all words (seen and unseen) occurring in the test data.

The smoothing scheme is implemented as follows. First, the test corpus C is tagged with a standard part-of-speech tagger (We use Treetagger (Schmid, 1994)). Tokens of words w and POS tags τ in the corpus are tabulated to obtain a frequency table $c(w, \tau)$.

w	word
τ	tag
ι	incorporation sequence (consisting of multiple features)
$t(w, \tau, \iota)$	treebank frequency
$c(w, \tau)$	test corpus frequency
$c'(w, \tau)$	scaled test corpus frequency
$t_t(w, \tau, \iota)$	smoothed treebank frequency

First, frequencies in the test corpus $c(w, \tau)$ are scaled to be comparable to the size of treebank as shown in Equation 2.1

$$c'(w, \tau) = \frac{t(\tau)}{c(\tau)} c(w, \tau) \tag{2.1}$$

where $c(\tau)$ is the marginal frequency obtained by summing over w in the corpus. $t(\tau)$ is the marginal frequency obtained by summing over words w and incorporations ι (defined in Equations 2.5 and 2.6).

The scaled corpus frequency $c'(w, \tau)$ is then split among incorporations ι in proportion to the frequency of incorporations for the tag τ in the treebank, as shown in Equation 2.2. This is the same for novel and non-novel words for the reason described above.

$$c'(w, \tau, \iota) = \frac{t(\tau, \iota)}{t(\tau)} c'(w, \tau) \tag{2.2}$$

where $t(\tau, \iota)$ is the marginal frequency defined by summing over w (defined in Equation 2.4).

The corpus frequency $c'(w, \tau, \iota)$ is then merged with the treebank distribution in a linear combination of t and c' to give a smoothed model t_t used to parse the test corpus.

$$t_t(w, \tau, \iota) = (1 - \lambda) t(w, \tau, \iota) + \lambda c'(w, \tau, \iota) \tag{2.3}$$

where $0 < \lambda < 1$.

For novel words or word-tag combinations in C, the first term on the RHS in Equation 2.3 is zero. The value of λ can be parameterized by tag τ and incorporation ι, and also by frequency of occurrence f of words or tags.

2.9.1 Marginal frequencies

Marginal frequencies are defined by summation.

$$t(\tau, \iota) = \sum_w t(w, \tau, \iota). \tag{2.4}$$

$$t(\tau) = \sum_w \sum_\iota t(w, \tau, \iota). \tag{2.5}$$

$$c(\tau) = \sum_w c(w, \tau). \tag{2.6}$$

$$t(w,\tau) = \sum_{\iota} t(w,\tau,\iota)$$
$$t(w) = \sum_{\tau} t(w,\tau) \tag{2.7}$$

We verify that marginal frequencies in the smoothed model t_t are identical to the unsmoothed treebank model t.

Identity for τ marginal:

$$\begin{aligned} c'(\tau) &= \sum_{w} c'(w,\tau) \\ &= \sum_{w} \frac{t(\tau)}{c(\tau)} c(w,\tau) \\ &= \frac{t(\tau)}{c(\tau)} \sum_{w} c(w,\tau) \\ &= \frac{t(\tau)}{c(\tau)} c(\tau) \\ &= t(\tau) \end{aligned} \tag{2.8}$$

Identity for τ, i marginal:

$$\begin{aligned} c'(\tau,i) &= \sum_{w} c'(w,\tau,\iota) \\ &= \sum_{w} \frac{t(\tau,\iota)}{t(\tau)} c'(w,\tau) \\ &= \frac{t(\tau,\iota)}{t(\tau)} \sum_{w} c'(w,\tau) \\ &= \frac{t(\tau,\iota)}{t(\tau)} c'(\tau) \\ &= \frac{t(\tau,\iota)}{t(\tau)} t(\tau) \\ &= t(\tau,\iota) \end{aligned} \tag{2.9}$$

Marginal for $c'(w, \tau, \iota)$:

$$\begin{aligned}
\sum_{\iota} c'(w, \tau, \iota) &= \sum_{\iota} \frac{t(\tau, \iota)}{t(\tau)} c'(w, \tau) \\
&= \frac{c'(w, \tau)}{t(\tau)} \sum_{\iota} t(\tau, \iota) \\
&= \frac{c'(w, \tau)}{t(\tau)} t(\tau) \\
&= c'(w, \tau) \qquad (2.10)
\end{aligned}$$

Identity for marginal distribution $t_t(\tau, \iota)$:

$$\begin{aligned}
t_t(\tau, \iota) &= \sum_{w} t_t(w, \tau, \iota) \\
&= \sum_{w} (1 - \lambda) t(w, \tau, \iota) + \lambda c'(w, \tau, \iota) \\
&= (1 - \lambda) \sum_{w} t(w, \tau, \iota) + \lambda \sum_{w} c'(w, \tau, \iota) \\
&= (1 - \lambda) t(\tau, \iota) + \lambda c'(\tau, \iota) \\
&= (1 - \lambda) t(\tau, \iota) + \lambda t(\tau, \iota) \\
&= t(\tau, \iota) \qquad (2.11)
\end{aligned}$$

Thus the τ, ι marginal frequencies in the smoothed model are identical to the unsmoothed treebank model.

2.9.2 Example to illustrate the smoothing scheme

As described before, the smoothing scheme assigns all possible incorporation sequences to word-tag combinations from the test data. Novel word-tag combinations get a distribution over all incorporations, i.e., the average treebank distribution for that tag. Non-novel word-tag combinations are assigned a distribution over all incorporations that largely reflects the treebank distribution for that word-tag combination but with some

frequency assigned to novel incorporations. The idea is as follows: for words that have not been seen at all, or have not been seen with a particular POS tag, we have no information regarding the frequency distribution over the incorporation sequence for that tag. Therefore, we assign the average distribution calculated over all words in the treebank for that POS tag. For words that have been seen in the treebank training data before with a particular tag, we maintain their treebank distribution but assign a small frequency to unseen incorporations in order to cover the possibility that the word may occur in the test data with an incorporation that was not seen in the treebank training data.

The effects of this smoothing method on lexical entries of words in the test corpus is illustrated below by merging a dummy treebank lexicon with a dummy corpus.

Lexicon

The dummy lexicon *t* in Figure 2.20 has four entries (words) A, B, C, D with associated tag-incorporation frequencies as shown below, using the notation of the previous section. The tags in the lexicon are T_1, T_2 and T_3 while incorporation sequences on these tags are x and y.

The same lexicon is represented below (Figure 2.21) in the format in which our lexicons are represented (as text files). The incorporation y is chosen to be more common than the incorporation x – x and y are in a 1: 2 ratio for all tags T_1, T_2 and T_3. The tag T_1 also occurs with the incorporation z (in the entry for word B), but there are no words with z associated with tags T_2 and T_3.

$$
\begin{aligned}
t(A, T_1, x) &= 10 \\
t(A, T_1, y) &= 20 \\
t(B, T_1, x) &= 20 \\
t(B, T_1, y) &= 40 \\
t(B, T_1, z) &= 5 \\
t(C, T_1, x) &= 1 \\
t(C, T_2, x) &= 10 \\
t(C, T_2, y) &= 20 \\
t(D, T_3, x) &= 10 \\
t(D, T_3, y) &= 20
\end{aligned}
$$

Figure 2.20: A dummy treebank lexicon *t*.

Lexicon *t*			
A	T_1.x 10	T_1.y 20	
B	T_1.x 20	T_1.y 40	T_1.z 5
C	T_1.x 1	T_2.x 10	T_2.y 20
D	T_3.x 10	T_3.y 20	

Figure 2.21: Dummy Lexicon *t*

Corpus

The tagged test corpus C_1 (Figure 2.22) consists of one novel word N with tag T_1, and two occurrences of an existing word A. The word-tag combinations $c_1(N, T1)$ and $c_1(A, T_2)$ are novel in this corpus while the combination $c_1(A, T_1)$ is not.

Corpus C_1	
word	*tag*
A	T_1
A	T_2
N	T_1

Figure 2.22: Dummy Corpus C_1

Merged Lexicon: The corpus C_1 is merged with the lexicon using Equation 2.3, with $\lambda = 0.5$. The merged lexicon (Figure 2.23) now contains an entry for the novel word N. N occurs in the corpus only with tag T_1. Therefore, the entry for N will contain all incorporations that have occurred with T_1 in the treebank lexicon. The smoothed distribution for N is the average distribution of incorporations on T_1 in all words in the lexicon. For T_1, x, y, and z occur with a total frequency 31: 60: 5 in t; this ratio is maintained in the entry for N. For the non-novel word-tag combination $c(A, T_1)$, the treebank distribution for A is mostly maintained in the smoothed lexicon (with a small frequency added), but the entry for A now also contains T_1.z which was not seen with A in the treebank lexicon.

The test corpus also contains the novel word-tag combination $c_1(A, T_2)$. Thus, the entry for A in the merged lexicon also contains T_2 with all incorporations that T_2 has occurred with in the original lexicon, i.e $t(A, T_2, x)$ and $t(A, T_2, y)$, seen as T_2.x and T_2.y in the entry for A below. The relative frequencies of T_2.x and T_2.y are the average over the entire original lexicon t (approximately 1: 2, as stipulated for t).

Smoothed Lexicon t_t

A	T_1.x 10.0055	T_1.y 20.01	T_1.z 0.0025	T_2.x 0.01	T_2.y 0.02
B	T_1.x 19.98	T_1.y 39.96	T_1.z 4.995		
C	T_1.x 0.999	T_2.x 9.99	T_2.y 19.98		
D	T_3.x 10.0	T_3.y 20			
N	T_1.x 0.0155	T_1.y 0.03	T_1.z 0.0025		

Figure 2.23: Smoothed Lexicon

2.9.3 Determining the value of the smoothing parameter λ

The strategy of smoothing the lexical entries of both novel and non-novel corpus words would be justified only if the test data contained seen words with unseen incorporation sequences. It is possible that for high-frequency words in the treebank, this smoothing is not justified since these words have been seen with all likely incorporation sequences. We use verbs as the test case, and examine the distribution of tag and incorporation combinations for verbal categories in a held-out subset of the treebank corpus, as a factor of the occurrence frequency of the verb in the training corpus. Table 2.2 shows overall counts for (verb, tag, incorporation) tokens and types in the held-out corpus. More than 20% of (verb, tag, incorporation) combinations are novel (unseen) in the held-out corpus.

Table 2.2: Counts of Verbs with tags and incorporations in a held-out subset of the treebank corpus (approx. 4000 sentences).

	Tokens	Types	Novel Tokens	Novel Types
Verb.Tag.Inc	11999 (+688 ambiguous)	4329	898 (7.48%)	869 (20.07%)

Table 2.3 shows the token frequency for novel (verb, tag, incorporation) combinations in the held-out data, split according to the occurrence frequency of the verb in the training data. This data gives an idea of the effect of training occurrence frequency of a verb on the likelihood that it occurs in new data with a new (tag, incorporation) combination. As seen from Table 2.3, for low frequency verbs, there is a high percentage of types that are new in the new data. Even for high frequency verbs, almost 7% tag.incorporation types are unseen in the training data. However, these unseen types are a very small percentage of the overall novel tokens in the held-out data. This could means that while there are new frames in the new data for high frequency verbs, there is not much empirical gain in smoothing the entries of these words. In face, a smoothed distribution for such words might cause more harm than good. The experiments reported in this chapter use a constant value of the interpolation parameter λ which determines the weight given to the test corpus while smoothing. It might be beneficial to reduce the value of λ as the frequency of the lexical item in the training corpus increases, so that more weight is given to the treebank distribution for high frequency words. In order to determine the optimal value of this parameter empirically, further experimentation is required.

2.10 Evaluation of Treebank PCFG

In order to evaluate the performance of the treebank PCFG on parsing Wall Street Journal text, we report its performance on three measures: (i) the standard PARSEVAL measures of labeled bracketing recall and accuracy, (ii) detection of empty categories, and (iii) detection of correct verbal valence. The test procedure involves evaluating different versions of the PCFG corresponding to different combinations of features stipulated to be incorporated in the PCFG symbols. Below we present results for a PCFG with an

Table 2.3: Novel tag and incorporation sequences on verbs (subcategorization frames) in held-out data, split according to the frequency of occurrence of test verbs in the training data.

Verb Freq Range in Training Data **Verb**	% Novel Tokens in Test Data. **Verb Tag Inc**	%novel types in Test Data. **Verb Tag Inc**
0	100%	100%
1	48.7 %	53.30%
2	10.4 %	40 %
3	27.32 %	27.56%
4	18.24 %	20 %
5	19.44%	21.21%
6-10	12.31%	14.25%
11-15	13.13 %	15.5%
16-20	9.38 %	13.51%
21-30	5.97 %	8.9%
31-50	5.00 %	8.39%
51-100	3.60 %	7.72%
> 100	0.91 %	6.97%

optimal combination of features (based on a non-exhaustive testing of different feature combinations on the development data set).

2.10.1 Labeled Bracketing

We evaluate the quality of the PCFG extracted from the transformed treebank using standard PARSEVAL measures. Maximum probability (Viterbi) parses are obtained for all

Table 2.4: Labeled bracketing evaluation, Penn Treebank Section 23.

	Our best model	Schmid (2006)	Klein & Manning (2003)
Labeled Recall	86.5	86.3	85.1
Labeled Precision	86.7	86.9	86.3
Labeled F-score	86.6	86.6	85.7

Table 2.5: Labeled bracketing evaluation, for PCFGs with prepositions incorporated in verbal and nominal categories, Penn Treebank Section 23.

	Prepositions on verbs	Prepositions on nouns
Labeled Recall	86.11	85.98
Labeled Precision	86.50	86.3
Labeled F-score	86.31	86.14

sentences in the standard test section of the Penn Treebank (Section 23), using a given smoothed PCFG model and the parser Bitpar (Schmid, 2004). Table 2.4 shows the labeled bracketing scores for the best-performing PCFG model. The labeled bracketing scores are comparable to state-of-the-art unlexicalized grammars. The exact combination of features incorporated into treebank categories for this version of the PCFG is listed in Appendix A.

Incorporating different features into the PCFG changes the labeled bracketing score of the PCFG; our framework allows us to stipulate which features from the feature constraint grammar are to be incorporated into the symbols of the PCFG. To illustrate this point, consider features on verbs and nouns related to valence. In the grammar version whose scores are reported in Table 2.4, features incorporated on verb and noun categories do not include the specific preposition for prepositional subcategorization

frames. In another version of the grammar, we include specific prepositions from the prepositional complement into the verbal and nominal subcategorization frame. The effect of including specific prepositions on these categories may be to make the grammar too sparse, resulting in the reduction of the labeled bracketing score seen in Table 2.5. Nevertheless, including these features is interesting from the point of view of creation of lexical resources, since it enriches the lexical information that is represented. Figure 2.24 shows a Viterbi parse of a Penn Treebank sentence using the PCFG. Notice that each non-terminal in the tree consists of a Penn Treebank category symbol, followed by a sequence of incorporated features.

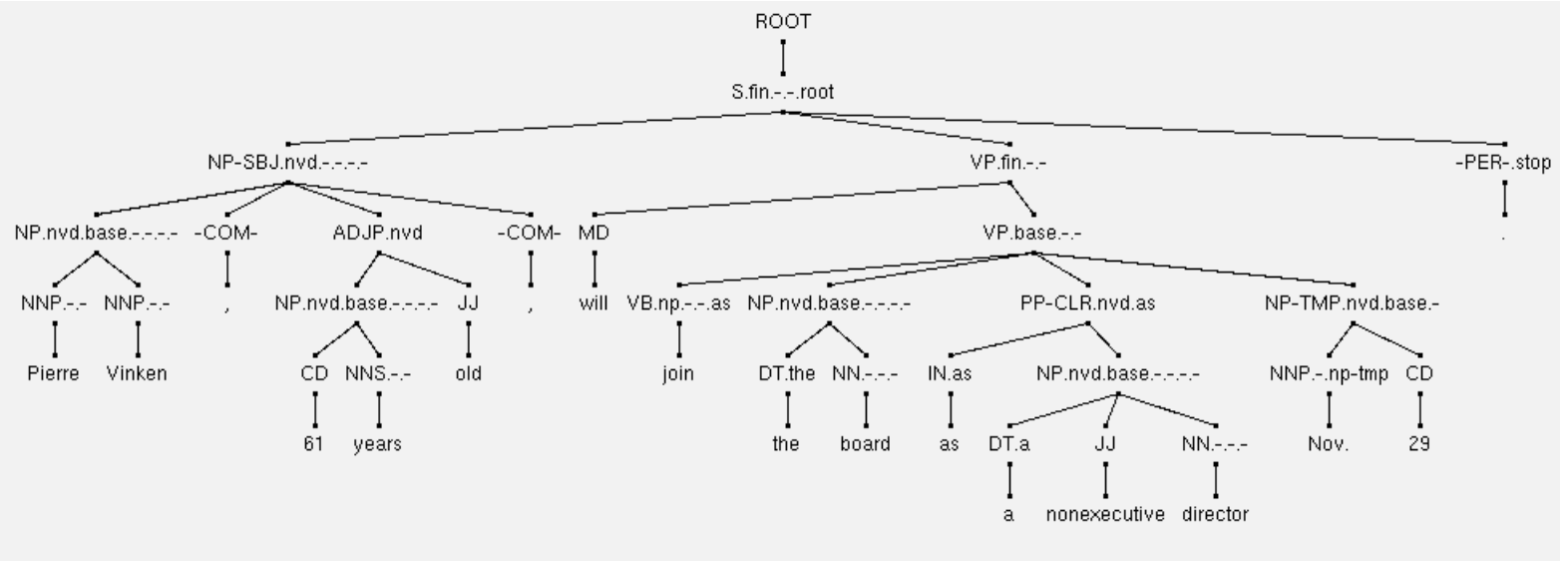


Figure 2.24: A Viterbi parse tree generated by the PCFG.

2.10.2 Empty Category Detection

We also report figures for detection of empty categories in Section 23 of the Penn Treebank. While our effort has not been directed specifically towards the task of empty category detection, the models do fairly well on this task. Table 2.6 shows precision and recall results on the task of empty category detection for all empty categories in Section 23. The evaluation of empty categories is done as follows: Viterbi parses are obtained for all sentences in Section 23. The Viterbi parses are compared to the gold standard treebank parses for precision and recall accuracy for empty categories. Table 2.6 also shows the best empty category detection results in the field at this time (Schmid, 2006). Schmid (2006) currently out-performs our model by only a small amount.

2.10.3 Valence Detection

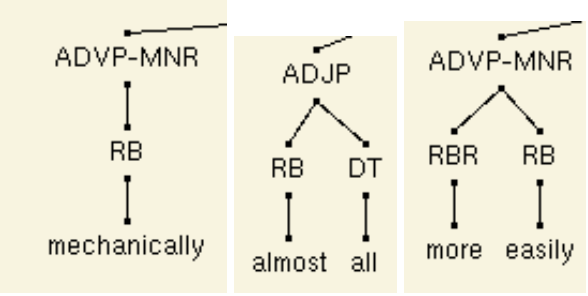
The third metric used to evaluate treebank models is the task of detecting correct subcategorization frames of verbs in Viterbi parses of a held-out portion of the Penn Treebank. The subcategorization frame of a verb is defined as the POS tag of the verb followed by the sequence of features incorporated into the verbal POS tag. In the PCFGs reported here, the features incorporated into verbal tags are `Val` (valence), `Vsel` and `Sbj`. We have a total of 81 subcategories of verbs. As before, we obtain Viterbi parses of all sentences in a held-out portion of the treebank. The pre-terminal category of all verbs in the Viterbi parses (its POS-tag and incorporation sequence) is compared to the pre-terminal tags in the transformed treebank (which forms the gold standard). The number of verb tokens in this testset is 6073. Total f-score in detecting the correct valence is 73.31 with recall accuracy being 74.08% and precision being 72.57%. See §4.5.3 for details of this evaluation, such as descriptions of subcategorization frames in the grammar.

Table 2.6: Empty category evaluation on section 23. Our empty category symbols are in the first column, with the corresponding Penn Treebank category shown in brackets. Following Schmid (2006), we only report empty categories which have an occurrence frequency of more than 6 in the test data. The last row shows results for all empty categories in section 23.

EC	Test freq	Precision	Recall	f-score	Schmid'06
E-NP (NP *)	1584	0.83	0.89	0.86	0.87
T-NP (NP *T*)	498	0.83	0.83	0.83	0.85
C (0)	406	0.94	0.90	0.92	0.92
U (*U*)	372	0.94	0.95	0.95	0.95
T-ADVP (ADVP *T*)	168	0.61	0.47	0.53	0.72
T-S (S *T*)	160	0.81	0.87	0.85	0.90
T-S-SBAR (SBAR-S *T*)	116	0.79	0.78	0.79	0.82
WHNP (WHNP 0)	49	0.80	0.80	0.80	0.60
ICH-PP (PP *ICH*)	29	0	0	0	0.05
T-PP (PP *T*)	28	0.69	0.39	0.50	0.58
EXP-SBAR (SBAR *EXP*)	16	0	0	0	0.17
EXP-S (S *EXP*)	14	-	0	0	0.69
ICH-S (S *ICH*)	13	-	0	0	0.59
ICH-SBAR (SBAR *ICH*)	13	-	0	0	0.35
WHADVP (WHADVP 0)	11	0.18	0.18	0.18	0.60
ELL-NP (NP *?*)	10	0	0	0	0.15
T-ADJP (ADJP *T*)	9	0.6	0.68	0.63	0.88
ELL-SBAR	8	-	0	0	-
T-VP (VP *T*)	8	-	0	0	0.55
ELL-VP (VP *?*)	7	0.38	0.43	0.4	0.36
total	3519	0.84	0.83	0.83	0.84
all ECs	3567	0.84	0.82	0.83	-

2.11 EM-based clustering of Local Syntactic Contexts of Words

The above sections described lexical features that encoded structural information such as subcategorization frames and attachment preferences associated with lexical items like verbs, adverb, nouns, etc. These features are "proposed" features in that they are added because insight about linguistic generalizations and phenomena informs us that they are likely to be useful; the set of their values were also determined based on common values seen in languages, and distinctions made in Penn Treebank II annotation. For example, the `advptype` feature takes three values based on the generalization that adverbs in English can be sub-classified as sentential, nominal or verbal adverbs. In this section, we describe another type of feature whose values are not proposed but are learnt automatically from the treebank data. This feature is a generic feature on pre-terminal categories which divides the pre-terminal category into sub-classes. The sub-classes are determined by clustering the local syntactic contexts in which the pre-terminal symbol occurs in the training sections of the treebank. The clustering algorithm used is EM. The motivation behind adding this class feature is to divide pre-terminal categories into finer categories by automatic classification. It is possible that there are sub-classes (based on patterns in their syntactic contexts) that are not being captured by the linguistic feature annotation described above but may be captured by automatic classification. Also, grammar development for some categories such as adjectives and adverbs is not yet complete: the EM class feature gives us an automatic and fast way of capturing some of the syntactic preferences of words belonging to these categories. We focus on pre-terminals due to our interest in lexical acquisition. There have have been previous successful attempts to automatically learn the most useful refinements for Treebank categories that will result in improved parsing results. For example, Prescher (2005) uses EM to deduce refinements for categories for the entire grammar.



mechanically	0	ADVP-MNR → RB
almost	0	ADJP → RB DT
easily	1	ADVP-MNR → RBR RB

Figure 2.25: Example of extracted local contexts for the category RB.

2.11.1 Method

For a given pre-terminal category, we collect from the training sections of the treebank all pairs of words of that category and their local syntactic context. The local syntactic context of a word consists of the immediate syntactic rule containing its pre-terminal category (both the LHS and RHS). As an example, Fig. 2.25 shows local tree configurations for three different adverbs (RB). Their local syntactic context consists of the rule that the pre-terminal occurs in, shown below the sub-trees as a tuple consisting of the word, the position of its pre-terminal on the RHS of the syntactic rule, and the syntactic rule itself.

EM based clustering can be seen as an estimation problem for a latent class model (Rooth et al., 1999). The incomplete data space is $\mathcal{Y} : (w, l)$ while the complete data space is $\mathcal{X} : (c, w, l)$.

Observed incomplete data sample space	$\mathcal{Y} : (w, l)$
Complete data sample space	$\mathcal{X} : (c, w, l)$
$w \in W$	word
$l \in L$	local syntactic context of word
$c \in C$	class

Classes corresponding to the pairs (w, l) are viewed as hidden variables or incomplete data for the EM clustering algorithm. The complete data space $\mathcal{X} : (c, w, l)$ is related to the observation y as $\mathcal{X}(y) = \{x \in \mathcal{X} \mid x = (c, y)\}$.

The complete data specification $P_\theta(x)$ corresponding to the joint probability $p(c, w, l)$ has the parameter vector $\theta =< \theta_c, \theta_w, \theta_l >$. The EM algorithm finds the value $\hat{\theta}$ of θ that maximizes the incomplete log likelihood $L(\theta)$.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, L(\theta) \quad \text{where} \quad L(\theta) = ln \prod_{y} P_\theta(y)$$

$P_\theta(x)$ is related to $p_\theta(y)$ as the marginal probability

$$\begin{aligned} P(w, l) &= \textstyle\sum_{c \in C} P(c, w, l) \\ &= \textstyle\sum_{c \in C} P(c)\, P(w, l \mid c) \\ &= \textstyle\sum_{c \in C} P(c)\, P(w \mid c) P(l \mid c) \end{aligned} \tag{2.12}$$

Note that w and c are conditioned not directly on each other, but only through c due to a conditional independence assumption.

For each syntactic context l in L a *unique* class $\bar{c}$ is then determined. This is the class in which the context has the highest frequency in the last EM iteration. Thus we obtain the pairs $(l, \bar{c})$.

2.11.2 Induced Classes

Using the above method, we induced unique classes for pre-terminal categories of open class lexical items (adjectives, adverbs, verbs, nouns), using Penn Treebank sections 0-22 as the training data, with some portion held out for testing. The clustering software was adapted from that used in (Rooth et al., 1999) (implemented by Mats Rooth).

Some induced classes are shown below for the past tense verb VBD, along with the highest probability local context associated with the class, and examples of some words that fall into the class.

1. Class 6 : **VP** → **VBD SBAR** (*said, added, noted, reported, announced, found, thought, estimated*)
2. Class 7 : **VP** → **VBD S** (*agreed, declined, began, called, continued, failed, had, wanted*)

Past tense verbs that take an SBAR complement seem to fall neatly into one class (Class 6). The top local context associated with this class is the VP rule with an SBAR complement. Class 7 is composed of verbs that select for an S complement.

Examples of induces classes for adverbs (RB), with the top local context, and some examples of adverbs in each class are shown below. Adverbs of manner seem to fall into one class (Class 6). Adverbs that function as particles, and which appear as heads of ADVP-CLR (an adverbial complement of verbs (for example used as *went up, went down*, etc.) fall into a distinct Class 2. Adverbs that modify an adjective fall into another class (Class 7). Thus, the induced classes seem interpretable linguistically to some extent.

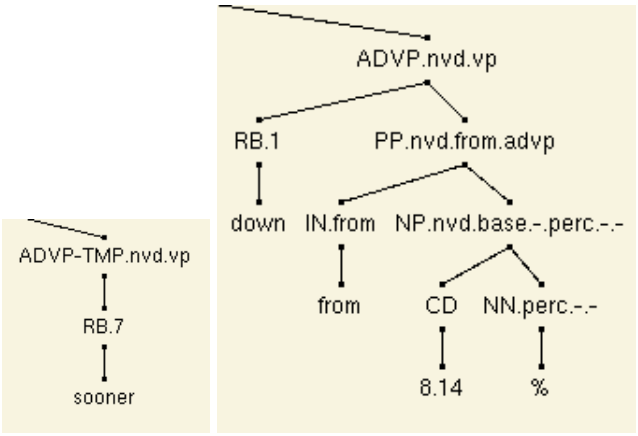
1. Class 6 **ADVP-MNR → RB** (*as, well, quickly, closely, sharply, slightly, directly, publicly*)
2. Class 2 **ADVP-CLR → RB** (*up, down, back, here , there, away, ahead, enough*)
3. Class 7 **ADJP → RB JJ** (*so, very, as, too, relatively, not, highly, almost*)

2.11.3 Adding the EM Class feature to the Treebank Feature-grammar

Each rule in the treebank that contains a pre-terminal node is associated with a maximum-probability class (as induced in the last section) of which it is a member. For each relevant rule in the feature grammar, an *EM-class* feature is added to the pre-terminal category, whose value is the maximum-probability class associated with that rule. The four classes that are added, and their associated values (indicating the number of classes) are shown below (each category need not have been given a feature with a different name, this was done just for clarity).

Category	*Name of feature*	*Possible values*
RB	`rbclass`	{0,1,2,3,4,5,6,7,-};
NN and NNS	`nclass`	{0,1,2,3,-};
JJ	`jjclass`	{0,1,2,3,-};
All verbs	`vclass`	{0,1,2,3,-};

The value of the EM-class feature is passed down into the lexicon, i.e. each word (for the above categories) gets annotated with the class value associated with its pre-terminal. The class feature is incorporated into the pre-terminal category while building



(a) (b)

Figure 2.26: Examples of tree-fragments parsed with the PCFG with an incorporated class feature on RB, for the adverbs *sooner* and *down*.

the PCFG. The motivation behind incorporating the class feature into the pre-terminal symbol of the PCFG is that it encodes the rule shape that is associated with a particular lexical item. Just like the subcategorization features, better values for this parameter can be learnt from large data using unsupervised methods. Examples of treebank trees with incorporated class features are shown in Fig 2.26 (sub-classes of the category RB are seen as RB.7 in (a) and RB.1 in (b)).

2.11.4 Evaluation

The efficacy of the EM-class feature was evaluated by two means:

- The labeled bracketing score of Viterbi parses of sentences from a held-out portion of the treebank, parsed with PCFG models with the EM-class features incorporated into lexical (pre-terminal) category symbol.
- Comparing the local syntactic context of all words of a given category in Viterbi

Table 2.7: Adding an EM-class feature on the adverb category RB improves labeled bracketing f-score

Version	a15	b1
tag	-	RB
# classes	0	8
Recall	85.91	86.56
Precision	86.42	86.78
FMeasure	86.17	86.67

parses of the test-set against the gold standard local contexts for those words obtained from the treebank.

The results for several grammar versions with EM-class features on different categories are given below. Grammar versions with a version name starting with *a* do not have a class feature incorporated and form the baseline. Version names starting with *b* have EM-class features on categories as indicated and are identical to the *a* baseline versions in all other respects.

Adding EM-class feature to the category RB

The RB (adverb) category was subdivided into 8 classes by adding an EM-class feature with 8 values. The baseline is the grammar version **a15** with no features on category RB. Adding the EM-class feature (8 classes) improved the labeled bracketing f-score, as shown in Table 2.7.

Table 2.8: Adding an EMclass feature to the category DT reduces f-score

Version	b1	b5
tag	RB	RB, DT
# classes	8	8,4
Recall	86.56	86.39
Precision	86.78	86.63
FMeasure	86.67	86.51

Adding EM-class feature to the category DT

Using the grammar with the EM-class feature on RB as a baseline (Version b1), an EM-class feature was added to the category DT. This reduced the f-score (Table 2.8)

Adding EM-class feature to the category NN

Using grammar version b1 as baseline, an EM-class feature was added to the noun category NN. The up feature on NN was retained. The f-score improves, as shown in Table 2.9 for 4 class values, and less so for 2 class values.

Adding the EM-class feature to the category NNS

Adding the EM-class feature to NNS reduces f-score (Table 2.10)

Table 2.9: Adding an EM-class feature to NN improves f-score in version b6 and b8. Dividing NN into 4 subcategories (version b6) is better than dividing it into 2 subcategories (version b8)

Version	b1	b6	b8
tag	RB	RB, NN	RB, NN
# classes	8	8,4	8,2
Recall	86.56	86.81	86.69
Precision	86.78	86.98	86.82
FMeasure	86.67	86.89	86.75

Table 2.10: Adding EM-class feature to NNS reduces f-score in version b7. The baseline is version b6 with the class feature only on NN

Version	b6	b7
tag	RB, NN	RB, NN, NNS
# classes	8,4	8,4,4
Recall	86.81	86.67
Precision	86.98	86.78
FMeasure	86.89	86.72

Adding the EM-class feature to category VBD

Adding an EM-class feature to the past tense verb category VBD while maintaining all other verbal features improves f-score very slightly. Removing the valence feature `Val` while adding the EM-class feature on VBD reduces the f-score, indicating that `Val` captures some regularities that the EM-class feature does not. Removing all features from VBD except for the EM class feature further reduces the f-score indicating that the original verbal features are useful. See Table 2.11.

Table 2.11: Comparison of features (valence and emclass) on VBD

Version	b1	b10	b10(no Val)	b10(only EMclass)
tag	RB	RB,VBD	RB, VBD	RB, VBD
# classes	8	8,4	8,4	8,4
Recall	86.56	86.57	86.51	86.39
Precision	86.78	86.79	86.75	86.65
FMeasure	86.67	86.68	86.63	86.52

Table 2.12: EM-class feature on JJ improves f-score

Version	b1	b11
tag	RB	RB, JJ
# classes	8	8,4
Recall	86.56	86.71
Precision	86.78	86.94
FMeasure	86.67	86.83

Adding the EM-class feature to category JJ

Adding the feature to the adjective category JJ results in a better f-score (Table 2.12). There are no other features on the category JJ.

2.11.5 Local Context Evaluation

We performed an additional evaluation of PCFG models with EM-class features by comparing the local syntactic context of each word of a given category in Viterbi parses of test sentences produced with that model, with the local context of those words in the

Table 2.13: Recall accuracy of local context detection in viterbi parses in held-out test-set, in comparison with local context of words in treebank gold standard trees. Precision accuracy is similar.

Version	a15	b1	b6	b10
EM-class on	-	RB	RB, NN	RB,VBD
RB	48.78	60.00	59.73	59.2
NN	81.7	81.86	81.81	81.83
VBD	65.03	65.98	65.74	66.67
overall	75.18	75.87	75.96	75.87

treebank gold standard. All terminals of a particular category like RB, NN or VBD were extracted from the test set along with their local syntactic context (consisting of the LHS and RHS of the rule to which the pre-terminal belonged). For each word, the local context in the Viterbi parse was compared to the local context of the word in the corresponding gold standard treebank tree. Table 2.13 shows the recall accuracy in detecting the correct local context on words belonging to categories RB, NN and VBD, in Viterbi parses obtained with models with the EM-class feature incorporated on RB, NN and VBD. The baseline for each of the *b* version grammars is the grammar version a15, which does not have the EM-class feature on any of these categories.

2.11.6 Obtaining a final treebank model: EM-class feature on RB, JJ, NN, NNS and all verb classes

Based on the above evaluations on models in which the EM-class feature was added to each category separately, this feature was seen to be beneficial for nominal, adverb, adjective, and the verb categories. We then created a model in which the EM-class fea-

Table 2.14: The f-score on held-out data on a version with the EM-class feature on RB, NN, NNS, JJ and all verbal categories

Version	a19	b14
Tags with EM-class feature	None	RB, NN, NNS, JJ, all verbs
Recall	86.13	86.68
Precision	86.54	86.71
FMeasure	86.34	86.69

ture was added to all these categories: RB, JJ, NN, NNS and all verbal categories (VB, VBN, VBD, VBG, VBP, VBZ). Adding the feature to category NNS resulted in a slight decrease in *f*-score as opposed to the category NN on which it was beneficial (see Table 2.10); however we decided to retain the feature on NNS for the sake of consistency. The version number of this grammar is **b14** and is based on the version **a19**, which has no EM-class features on any category. The version **b14** is trained on sections 0-22 of the PTB, with two testsets (Testset I and Testset II) held out for evaluation purposes. These testsets are described in detail in Chapter 4 §4.5.3. The labeled bracketing f-score for model **b14** shows an improvement over the corresponding model without EM-class features, as shown in Table 2.14. This model (version b14) is frozen as the model to be used in the experiments on unsupervised estimation, described in the rest of this thesis. Version **b14** does not contain specific prepositions incorporated into verbal or nominal symbols. We also obtain a model **b15** that is identical to **b14** except for having prepositions incorporated into all verb categories and the two nominal categories NN and NNS.

2.12 Chapter Summary

This chapter described in detail the construction of an accurate unlexicalized PCFG from the Penn Treebank. We use a method based on constraint-solving to automatically annotated Penn Treebank trees with feature-structures. Features are then incorporated in the symbols of a context-free grammar and frequencies are collected, resulting in a probabilistic grammar and a probabilistic lexicon which encodes lexico-syntactic features. The PCFG contains very few features as compared to, say, the unlexicalized PCFG in Klein & Manning (2003). We have a total of only 19 features. The performance of the PCFG is fairly good; we expect that a horizontal markovization of long right hand sides of rules, and further grammar development will improve performance. The PCFG was evaluated on three counts: labeled bracketing, detection of empty categories and detection of verbal subcategorization frames. Addition of a class feature to subcategorize pre-terminal categories, based on EM clustering of local syntactic contexts of pre-terminals was described and shown to be effective. The treebank lexicon is smoothed before parsing new text to include entries for new word and tag combinations and assign them frequency, as well as to smooth existing entries. The smoothing method was described. The final version of the PCFG (version b14) described in this chapter is used in the Inside-outside re-estimation experiments described in the following chapters.

The treebank-annotation framework described in this chapter was developed from the point of view of creating feature-constraint grammars for languages other than English which have an existing treebank resource, the automatic annotation of these treebank with features, and construction of finer PCFGs for these languages. The grammar development environment is a comfortable one, and the existence of a treebank nearly eliminates the issue of ambiguity while developing the feature-constraint grammar. It would have been possible to use a simpler method for adding the features described in

this chapter to the Penn Treebank trees (such as by using a program to directly read in a tree and add appropriate features to its nodes). However, this framework is justified by its potential utility for developing feature-constraint grammars and treebank-PCFGs for other languages.

CHAPTER 3

UNSUPERVISED LEARNING OF SYNTAX: THE INSIDE-OUTSIDE ALGORITHM

3.1 Introduction

The inside-outside algorithm is an instance of the Expectation Maximization (EM) algorithm, and is the mainstay of unsupervised learning algorithms for PCFGs. Baker (1979) generalized the parameter estimation method for Hidden Markov Models (HMMs) to PCFGs as the inside-outside algorithm. Its modern form was invented by Lari & Young (1990) when they generalized the inside-outside algorithm of Baker (1979) to allow the training corpus to contain arbitrarily large number of sentences (Prescher, 2005). In this chapter we describe a procedure for the unsupervised learning of PCFG parameters that is based on the inside-outside algorithm. Inside-outside is an iterative procedure that, given an initial model, re-estimates the model parameters by maximizing the likelihood of a training corpus at each iteration. The algorithm can be used to induce model parameters from scratch or to re-estimate parameters of a model obtained by other means. We describe a method in which we re-estimate the parameters of a smoothed treebank PCFG. Secondly, we modify the procedure so that some parameters, namely the syntactic parameters, are "fixed", and the rest, namely the lexical parameters, are re-estimated. The motivation arises from the nature of natural language data, in that information related to specific lexical items is sparse and hence difficult to estimate from annotated data. In this chapter, we first describes the standard procedure for inside-outside estimation of PCFGs (§3.2), followed by a discussion of previous research related to learning grammars with inside-outside or its variants (§3.4). We then present the motivation and a theoretical description of our procedure.

The EM algorithm was defined by Dempster et al. (1977). It is an iterative parameter-estimation method used in settings where it is difficult to use an estimation method like maximum-likelihood estimation directly due to part of the data being "hidden" or "incomplete". The trick of the EM algorithm is to map the given *incomplete* training data to *complete* data, on which it is known how to perform maximum-likelihood estimation. At each iteration of EM, the likelihood of the observed i.e. the incomplete data is guaranteed to be non-decreasing. The EM algorithm is used when the following holds true (from Prescher (2005)):

- Direct maximum-likelihood estimation of a given probability model on the incomplete data is not possible, but maximum-likelihood estimation on the complete data is easy.
- There is an obvious (albeit ambiguous, that is, one-to-many) mapping from incomplete to complete data. The incomplete corpus can be considered to be mapped to the complete corpus by means of a *symbolic analyzer* that assigns to each incomplete data type a set of complete data types. The ambiguity in the mapping between an incomplete data type to its set of complete data types is resolved by means of a *statistical analyzer*.

The estimation of grammars from a corpus of raw text is a problem that fits well into the group of problems for which EM is expected to be useful. The incomplete data are the set of unlabeled corpus sentences, while the complete data are the parse trees associated with the corpus sentences. Since each sentence typically has multiple analyses (or parses), the mapping between the incomplete and complete data is ambiguous and is done by means of a *statistical parser*, which assigns to each incomplete data type a set of parse trees with probabilities associated with them.

3.1.1 Estimating PCFGs with Inside-outside

There has been a long history of research on learning PCFGs using the inside-outside algorithm. At the end of it, the general consensus in the field is that inside-outside works poorly on raw language text. There are several reasons for this: one is that PCFGs have a large number of parameters and the solution space of the likelihood function contains a large number of local maxima. The algorithm converges on one of these local maxima, without ever reaching the global maximum. Secondly, the algorithm is sensitive to starting conditions, and will converge on different solutions for different starting models (de Marcken, 1995). Thirdly, linguistic theory and conventions would have it that sentences be given certain structures that are interpretable in a manner consistent with the theory. Models induced from raw text will not necessarily conform to these conventions, which makes the interpretation and evaluation of models induced by inside-outside a difficult and subjective task. This issue of learnt models not resulting in linguistically desirable structures is not specific to the inside-outside algorithm, but in general is common to inference methods that are based on information-theoretic criteria. For example, there are some word sequences that occur very commonly in natural language, yet are not constituents of the same bracketed structure. A common example in English is the subject pronoun and auxiliary sequence such as *I am* , or a verb and preposition sequence such as *give to*. Since words in these sequences co-occur very frequently, methods based on mutual information will group them together as constituents, even though linguistic theory does not treat them as such. This problem has been addressed in some research to a certain extent (for example, in Magerman & Marcus (1990)), but only in an ad-hoc manner such as by stipulating lists of categories that can never be constituents. We will revisit the above issues in more detail while discussing previous research related to inside-outside estimation of PCFGs. The modified re-estimation procedure described here addresses some of them.

3.2 The EM Algorithm

In this section, the EM algorithm is presented briefly, followed by a description of PCFG-estimation. We follow the notation and explanation used in Prescher's (2005) tutorial on EM. The expectation and maximization steps in PCFG estimation are related to the treebank-training method for PCFGs, which leads to a simple and intuitive explanation. Prescher (2005) contains the proofs for relating the EM algorithm to treebank-training.

First, some definitions are given below.

Probability Distribution

A function $p : \mathcal{X} \rightarrow \mathcal{R}$ is a probability distribution on $\mathcal{X}$, where $\mathcal{X}$ is a countable set of types, if

$$p(x) \geq 0 \quad \forall\ x \in \mathcal{X}$$

$$\sum_{x \in \mathcal{X}} p(x) = 1$$

Probability Model

A probability model $\mathcal{M}$ on a set of types $\mathcal{X}$ is a non-empty set of probability distributions on $\mathcal{X}$. The elements of $\mathcal{M}$ are called *instances* of the model $\mathcal{M}$. The *unrestricted* probability model is the set $\mathcal{M}'(\mathcal{X})$ of all probability distributions on the set of types $\mathcal{X}$.

Corpus

A real valued function $f : \mathcal{X} \rightarrow \mathcal{R}$ is called a corpus, if f's values are non-negative numbers. Each $x \in \mathcal{X}$ is a type, and each value of f is a type frequency.

$$f(x) \geq 0 \quad \forall \; x \in \mathcal{X}$$

The corpus size is given by

$$|f| = \sum_{x \in \mathcal{X}} f(x)$$

Maximum likelihood estimation

Maximum-likelihood estimation is a widely-used estimation method. It aims at selecting an instance of a given probability model $\mathcal{M}$ which might have generated a given corpus f.

The probability of a corpus f allocated by a distribution p, an instance of $\mathcal{M}$, is

$$L(f; p) = \prod_{x \in \mathcal{X}} p(x)^{f(x)}$$

$\hat{p}$ is the maximum likelihood estimate of $\mathcal{M}$ on f if and only if the corpus f is allocated a maximum probability by $\hat{p}$

$$L(f; \hat{p}) = \max_{p \in \mathcal{M}} L(f; p)$$

$$\hat{p} = \operatorname*{argmax}_{p \in \mathcal{M}} L(f; p)$$

3.2.1 EM

The EM algorithm is used to find estimates of local maxima by a maximum-likelihood criteria in cases where it is not possible to directly obtain the maximum-likelihood es-

timate of a model on a given corpus. Given a probability model, maximum-likelihood aims at finding the unknown distribution that might have generated the given corpus. When an annotated corpus is available, we are in a position of being "almost completely informed" (Prescher, 2005) since only the specific probability distribution that generated the corpus is unknown. The EM algorithm is used in settings where we do not have complete information regarding the corpus. Thus, instead of a *complete* data corpus, the input of the EM algorithm is an *incomplete* data corpus. There is also defined an ambiguous mapping from the incomplete data to the complete data. The EM algorithm iteratively maximizes the likelihood of the incomplete data corpus by maximum-likelihood estimation over a complete data corpus and converges to parameter values at a local maximum of the likelihood function. These concepts are defined below and the iterative procedure of EM is stated.

Incomplete and Complete data

Let $\mathcal{Y}$ be the set of incomplete data types and $\mathcal{X}$ be the set of complete data types. A function $\mathcal{A}$ that maps the incomplete data to complete data is called a symbolic analyzer. The set of analyses $\mathcal{A}(y) \subset \mathcal{X}$ are pairwise disjoint and the union of all sets of analyses $\mathcal{A}(y)$ is complete.

$$\mathcal{A} : \mathcal{Y} \rightarrow 2^{\mathcal{X}}$$

$$\mathcal{X} = \sum_{y \in \mathcal{Y}} \mathcal{A}(y)$$

For each $x \in \mathcal{X}$ there exists a unique $y \in \mathcal{Y}$ which is called the yield of x such that x is an analysis of y.

$$y = yield(x) \quad \text{if and only if} \quad x \in \mathcal{A}(y)$$

Statistical Analyzer

A pair $< \mathcal{A}, p >$ where $\mathcal{A}$ is a symbolic analyzer and p a probability distribution on the complete data types $\mathcal{X}$ is called a statistical analyzer. The symbolic analyzer $\mathcal{A}$ is a function which assigns a set of analyses $\mathcal{A}(y) \subseteq \mathcal{X}$ to each incomplete data type $y \in \mathcal{Y}$.

$$\mathcal{X} = \sum_{y \in \mathcal{Y}} \mathcal{A}(y)$$

The statistical analyzer resolves the ambiguity of the incomplete data types $y \in \mathcal{Y}$, by using conditional probabilities of the analyses $x \in \mathcal{A}(\mathrm{y})$

$$p(x \mid y) := \frac{p(x)}{p(y)}$$

The statistical analyzer is used to induce probabilities for the incomplete data types $y \in \mathcal{Y}$

$$p(y) := \sum_{x \in \mathcal{A}(y)} p(x)$$

The Complete-data Model

In addition to the incomplete data corpus and the statistical analyzer, each iteration of the EM algorithm requires a probability model on the complete corpus. This is the complete-data model $\mathcal{M}$, of which each instance p is a probability distribution on the complete-data types. A starting instance p_0 of the complete data model $\mathcal{M}$ is used in the first iteration. The EM algorithm then proceeds in two steps for each iteration: In the Expectation-step (E-step), the complete data corpus expected by the distribution q is computed, where q is the instance of the model $\mathcal{M}$ used in the current iteration. In the Maximization-step (M-step), the maximum-likelihood estimate $\hat{p}$ of $\mathcal{M}$ on the complete data corpus is computed.

1. for each $i = 1, 2, 3, \ldots$ do

2. $q := p_{i-1}$

3. E-step: compute the complete data corpus $f_q : \mathcal{X} \rightarrow \mathcal{R}$ expected by q.

   $f_q(x) := f(y) \,.\, q(x \mid y)$ *where* $y = yield(x)$

4. M-step: compute a maximum-likelihood estimate $\hat{p}$ of $\mathcal{M}$ on f_q.

   $L(f_q; \hat{p}) = \max L(f_q, p)$

5. $p_i = \hat{p}$

6. end // for each i

7. output $p_0, p_1, p_2...$

The convergence properties of the EM algorithm guarantee that $L(f; p_0) \leq L(f; p_1) \leq L(f; p_2)...$

3.2.2 Induction of PCFGs

PCFG

A PCFG can be defined as a pair $< G, p >$ where G is a context free grammar, and p is the probability distribution over the set of all finite full-parse trees of G.

For all parse trees $x \in \mathcal{X}$,

$$p(x) = \prod_{r \in G} p(r)^{f_r(x)}$$

where r is a rule in G, p(r) is the rule probability, and $f_r(x)$ is the frequency of r in the parse tree x.

$\sum_{r \in G_A} p(r) = 1$ for all G_A, where G_A is a partition of the context-free grammar G with left-hand-side A.

$$G_A = \{r \in G \mid lhs(r) = A\}$$

Thus, a PCFG is defined by a context-free grammar G and some probability distributions on the grammar fragments G_A, thereby inducing a probability distribution on the set of all full-parse trees.

Treebank PCFG

For a given non-empty and finite corpus of full-parse trees (a *treebank*), the treebank grammar is the PCFG $< G, p >$ where G is the context-free grammar read off the tree-bank, and p is the probability distribution over the set of all finite full-parse trees of G induced by the following specific probability distributions on the grammar fragments G_A:

$$p(r) = \frac{f(r)}{\sum_{r \in G_A} f(r)} \tag{3.1}$$

where $f(r)$ is the number of times a rule r occurs in the treebank.

Relation between maximum-likelihood estimation of PCFGs and treebank PCFGs

It is well-known that each treebank grammar $< G, p >$ is the unique maximum-likelihood estimate of G's probability model on the given treebank (stated as Theorem 10 in Prescher (2005); Prescher (2005) also contains a simple proof). Thus it is guaranteed that p is a probability distribution on the set of full-parse trees of G, and that $< G, p >$ is a PCFG as in the definition above.

Theorem 10 (Prescher, 2005): Let $f_\tau : \mathcal{X} \rightarrow \mathcal{R}$ be a non-empty and finite corpus of full-parse trees, and let $< G, p_\tau >$ be the treebank grammar read off from f_τ. Then, p_τ is the maximum-likelihood estimate of $\mathcal{M}_G$ on f_T, i.e.

$$L(f_\tau; p_\tau) = \max_{p \in \mathcal{M}_G} L(f_\tau; p)$$

where $\mathcal{M}_G$ is the probability model of G defined by

$$\mathcal{M}_G = \left\{ p \in \mathcal{M}'(\mathcal{X}) \;\middle|\; p(x) = \prod_{r \in G} p(r)^{f_r(x)} \text{ with } \sum_{r \in G_A} p(r) = 1 \text{ for all grammar fragments } G_A \right\}$$

Prescher (2005) shows that the probability of the corpus f_T is given by

$$L(f_\tau; p_\tau) = \prod_A L(f_A; p)$$

Here, f_A is corpus of rules, read off from the given treebank, containing all rules with the left hand side A.

$$f_A(r) = f(r) \quad \text{if } r \in G_A$$

$$= 0 \quad \text{otherwise}$$

Maximizing $L(f_\tau; p_\tau)$ is equivalent to individually maximizing $L(f_A; p)$ for all A. Thus, the task is now to find the maximum-likelihood estimates of the model G_A on the corpus of rules f_A. The maximum-likelihood estimates can be shown to be the relative frequency estimates, which are easily calculated from f_A.

$$\hat{p}_A(r) = \frac{f_A(r)}{|f_A|} = \frac{f(r)}{\sum_{r \in G_A} f(r)}$$

Thus, from Eq. 3.1, for non-terminal symbols A

$$\hat{p}_A(r) = p_\tau(r) \quad \forall r \in G_A$$

Thus, the treebank grammar p_τ is the maximum-likelihood estimate of the probability model $\mathcal{M}_G$ on f_τ.

$$L(f_\tau; p_\tau) = \max_{p \in \mathcal{M}_G} L(f_\tau; p)$$

EM for PCFGs

Thus the procedure of the EM algorithm for PCFGs can be stated in terms of generating a treebank and reading the relative-frequency estimates off the treebank, as follows:

1. for each $i = 1, 2, 3, \ldots$ do
2. $q := p_{i-1}$
3. E-step (PCFGs) : generate the treebank $f_{\tau_q} : \mathcal{X} \rightarrow \mathcal{R}$ defined by

   $f_{\tau_q}(x) := f(y).q(x \mid y)$ *where* $y = yield(x)$
4. M-step (PCFGs) : read off the treebank grammar $< G, p_{\tau_q} >$
5. $p_i = p_{\tau_q}$
6. end // for each i
7. output $p_0, p_1, p_2 \ldots$

E-step: In order to obtain the complete corpus frequency f_{τ_q}, the frequency $f(y)$ of the incomplete corpus (i.e. of a training sentence) is distributed among the complete data types $x \in \mathcal{A}(y)$ according to the conditional probabilities $q(x \mid y)$. The conditional probability is calculated as follows:

$$q(x \mid y) := \frac{q(x)}{q(y)} \quad \text{where } y = yield(x)$$

The incomplete-data corpus can be recovered from the complete data corpus $f_q(x)$ by summing up all the frequencies $f_q(x)$ with $x \in \mathcal{A}(y)$. Thus, the conditional probability is

$$q(x \mid y) := \frac{q(x)}{\sum_{x \in \mathcal{A}(y)} q(x)} \quad \text{where } y = yield(x)$$

The complete corpus then is

$$f_{\tau_q}(x) := f(y).\frac{q(x)}{\sum_{x \in \mathcal{A}(y)} q(x)} \quad \text{where } y = yield(x)$$

These expectations are computed by the inside-outside algorithm with possible analyses being represented in a parse-forest representation.

M-step: In the M-step, the treebank grammar $< G_q, p_{\tau_q} >$ is read off the complete data corpus. In a treebank of hand-annotated sentences, each full-parse tree has a whole number frequency since a sentence is given a unique analysis. In the case of the complete data corpus, this frequency is a non-negative real number. G_q is the context-free backbone that is a subset of the original context-free backbone $G_q \subset G$.

p_{τ_q} is obtained from the relative frequency estimates of each subsection G_{qA} of the grammar.

$$p(r) = \frac{f(r)}{\sum_{r \in G_{qA}} f(r)}$$

3.3 How does EM learn grammars from ambiguous data: An example

Although sentences in the incomplete corpus typically have a large number of analyses, the EM algorithm can in principle learn to disambiguate them. This is because the corpus contains some structures that are unambiguous: it is from these unambiguous examples that it is possible to learn to disambiguate the ambiguous ones. It is well-known that, in order to be able to parse data accurately with treebank PCFGs, the context-free assumptions of standard treebank annotation have to be weakened. Similarly, it is im-

portant that the context-free backbone of the statistical analyzer (parser) used to obtain complete data during inside-outside estimation have some good properties which will result in probabilistic models that can resolve ambiguities. Previous research (de Marcken, 1995) has also pointed out that the form of the context-free symbolic backbone is important when trying to learn grammars with inside-outside. Below, we provide an example illustrating how EM learns to disambiguate from unambiguous examples, given a symbolic backbone that has suitable properties.

Consider the two sentences (1) and (2) below:

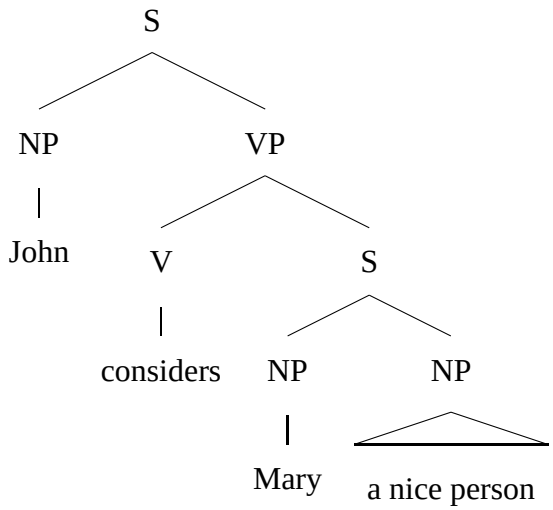
(1) John considers Mary a nice person.

(2) John gave Mary a nice present.

The complement of the verb *consider* in Sentence (1) is called a *small clause*, and is given an analysis in which the verb takes a clausal complement S which does not contain a VP (hence the term *small* clause), both in the Penn Treebank II annotation scheme, and according to some linguistic theories of small clauses (e.g. Stowell (1981)). The structure is shown in (3)[1].

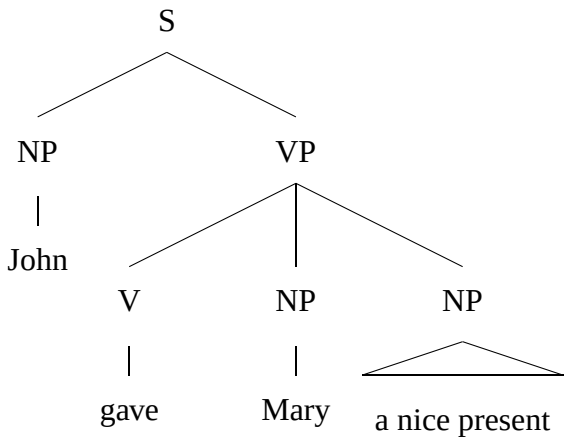
[1]The structure is simplified a bit to make the corresponding grammar smaller.

(3)



Sentence (2) contains the ditransitive verb *gave*, which takes two NPs as complements. The structure is shown in (4).

(4)



The context-free grammar required to construct these trees is given below in Figure 3.1 (some categories are not expanded fully).

Since the surface form of the two sentences (1) and (2) is the same (that is, they both have two NPs following the verb), they will have two analyses each which will have to be disambiguated: the small clause and the ditransitive analysis. The two analyses of

S → NP VP

VP → V S

VP → V NP NP

V → consider

V → gave

NP → Mary

NP → John

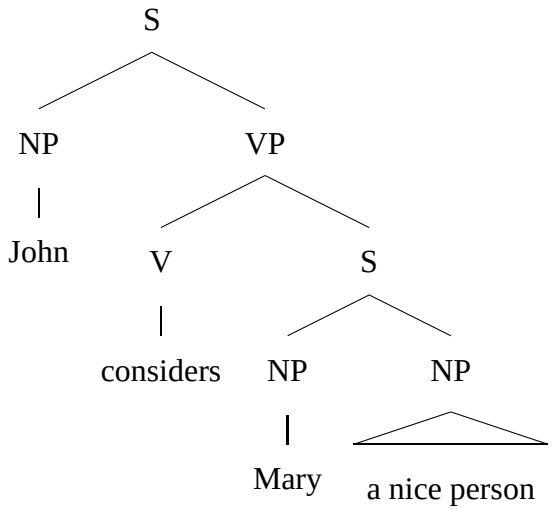
NP → a nice person

NP → a nice present

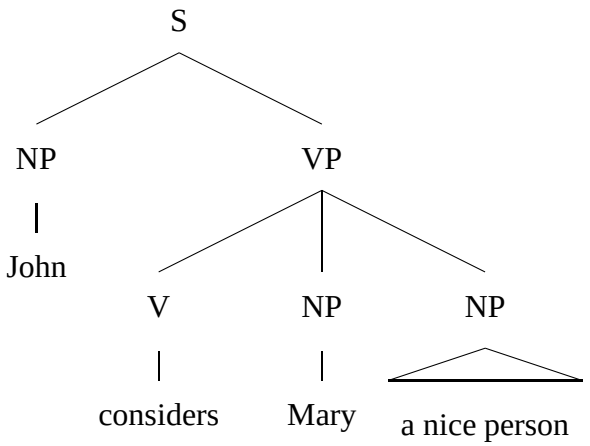
Figure 3.1: Context free grammar for a corpus of two sentences (3) and (4)

Sentence (1) are shown in (5-a) and (5-b). Similarly, (2) will also have two analyses.

(5) a.



b.



In general, in a probabilistic grammar, ambiguities are resolved by calculating all full-parse trees of a given sentence licensed by the context-free backbone of the PCFG, allocating probabilities to these trees using the rule probabilities of the PCFG, and choosing the tree with the highest probability as the best parse. During inside-outside estimation, the frequency of a training sentence will be distributed amongst its different analyses (full-parse trees) in proportion to their tree probabilities.

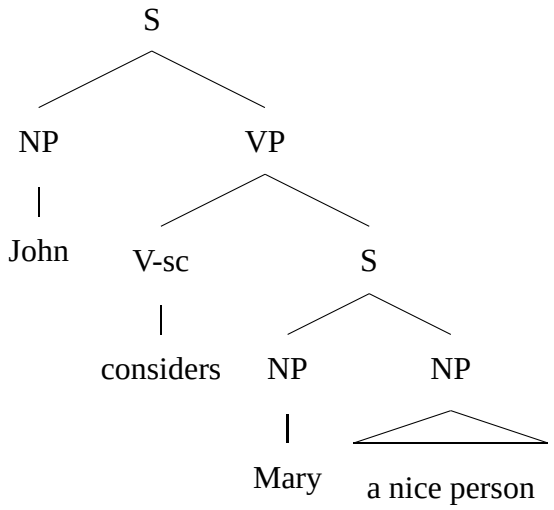
Thus, the frequency of Sentence (1) will be distributed among the two analyses (5-a) and (5-a) in the proportion of probabilities of rules used in the respective trees:

$$\frac{p(VP \rightarrow V\ S)\,.\,p(S \rightarrow NP\ NP)}{p(VP \rightarrow NP\ NP)}$$

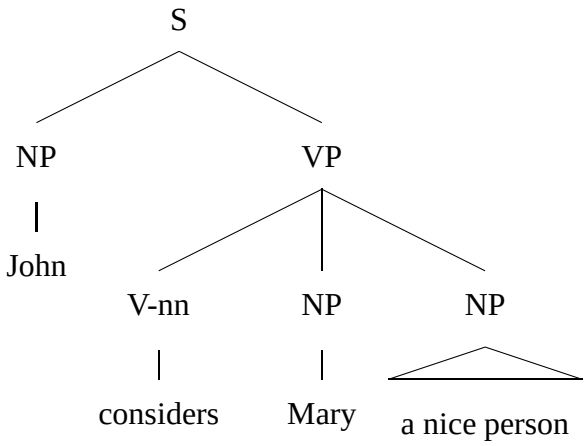
If $p(VP \rightarrow V\ S)\,.\,p(S \rightarrow NP\ NP) > p(VP \rightarrow NP\ NP)$, then the small clause analysis will get a higher frequency than the ditransitive analysis. The two analyses of Sentence (2) will be assigned frequencies in the same ratio. Thus, the two analyses of each sentence will have the same relative frequencies, although the small clause analysis is more appropriate for Sentence (1) and the ditransitive for Sentence (2).

Now consider the following scenario: since the small clause and ditransitive complements are selected by the respective verbs *consider* and *gave*, we create a grammar in which the complement is marked on the verbal pre-terminal verbal category V, thus dividing it into two subcategories: a small clause verb category and a ditransitive verb category. The two analyses for sentence (1) would then be as shown in (6), and for sentence (2) as shown in (7), still without a way to choose between them. The context free rules of the grammar associated with these trees are as shown in Figure 3.2.

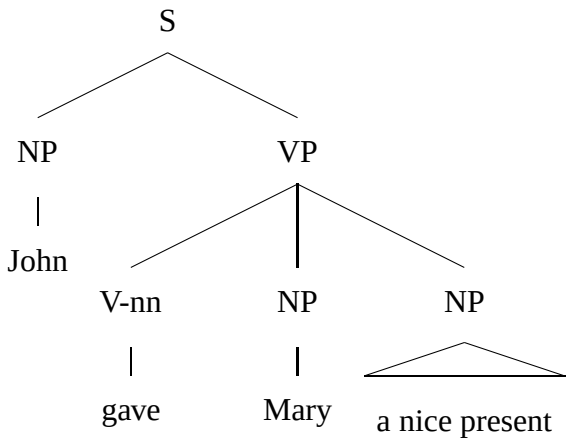
(6) a.



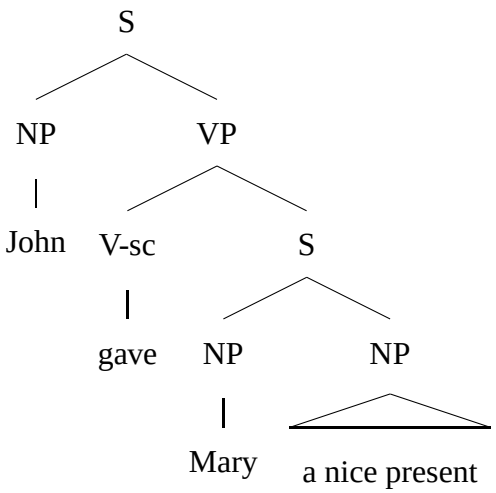
b.



(7) a.



b.



Now imagine that there is another sentence in the training data containing an occurrence of the verb *consider* with a small clause complement, but with a different type of small clause. For example, it may occur with an adjectival small clause complement as show in (8-a). (8-a) has a small clause analysis with an adjective predicate as shown in (8-b).

The following rule will be added to the previous grammar: S → NP ADJP.

(8-a) has an unambiguous analysis according to our grammar. Thus, the complete data corpus corresponding to these three sentences consists of five trees.

S → NP VP

VP → V S

VP → V NP NP

V-sc → consider

V-nn → consider

V-sc → gave

V-nn → gave

NP → Mary

NP → John

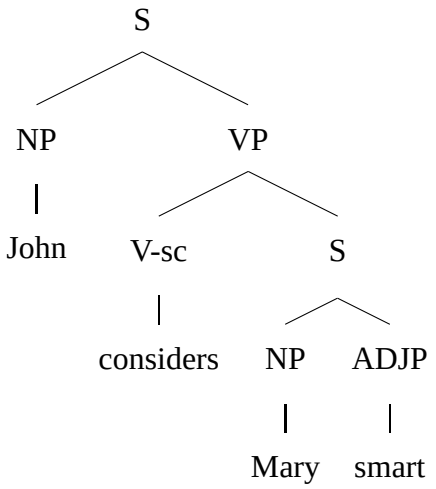
NP → a nice person

NP → a nice present

Figure 3.2: Context free rules for a finer grammar.

(8) a. John considers Mary smart.

b.



The rule "VP → V-sc S" occurs in three of five trees, and the rule "VP → V-nn NP NP" occurs in two trees. The rule "V-sc → consider" appears in three trees, while the rule "V-nn → consider" appears in two trees in the first iteration. In the second

iteration, due to the higher count of "V-sc → consider", the tree containing this rule will be more likely than the tree containing "V-nn → consider", for sentence (1). Therefore the frequency of the correct (small clause) analysis will be higher than the frequency of the wrong analysis for Sentence (1) in the second iteration, a desirable result. Notice that the rule "V-nn → gave" is not affected; hence either analysis for Sentence (2) is possible.

To summarize, the algorithm learns to disambiguate sentence (1) (the nominal small clause) due to the presence of sentence (8-a) (the adjectival small clause), which has an unambiguous analysis .

The above example illustrates the importance of having a context free grammar with the right properties. Consider a scenario in which the pre-terminal symbols for verbs in the example grammar do not distinguish the small clause and the ditransitive contexts. In training data consisting of the three sentences (1), (2) and (8-a), the frequency of the rule "VP → V S" will increase since it occurs in five trees, while "VP → NP NP" occurs in only two trees. However, this will make the small clause analysis more likely for sentence (2) as well as for (1), which is not desirable. Thus, this grammar will not be able to improve from the unambiguous occurrence of *consider* with an adjectival small clause.

Consider a second scenario in which the grammar marks not only a small clause and ditransitive context on verbs, but also distinguishes an adjectival small clause complement from a nominal small clause complement. The relevant rules in this grammar will look as follows:

VP → V-sc-nom S

VP → V-sc-adj S

V-sc-nom → consider

V-sc-adj → consider

In this case, the analysis of the sentence with the adjectival small clause ((8-a)) and the analyses of (1) will not have the same verbal rule. The two rules are "V-sc-adj → consider" and "V-sc-nom → consider" respectively. Hence the presence of the unambiguous sentence in the training data will not affect the relative frequencies of the two analyses of the ambiguous sentence (1).

The above examples illustrate that the algorithm will not be able to disambiguate sentence (1) when the given grammar is either too coarse (no complements marked on verbal category) or too fine-grained (types of small clauses are distinguished). Thus, we have seen how EM can learn to disambiguate ambiguous structures, but only given that representations in the context-free grammar are of a suitable nature.

3.4 Previous experiments in unsupervised estimation of PCFGs

Previous experiments in grammar induction using the inside-outside algorithm have led to the general opinion that inside-outside does not result in good grammars, although there have been some positive results as well (such as Carroll & Rooth (1998) and Beil et al. (1999)). De Marcken (1995) examines why previous approaches in unsupervised grammar acquisition did not succeed, and gives a detailed analysis of why inside-outside estimation did not produce plausible linguistic structures. Using an artificial corpus of three word sentences with a constant part-of-speech pattern, and a small lexicon, he demonstrates several properties of the grammar induction algorithm. The word co-

occurrences in the corpus are organized so as to emulate the distribution of verbs and prepositional phrases in a natural language corpus – there are strong dependencies between verbs and prepositions in following PPs, but not between prepositions and nouns within a PP. One result is that even with such simple sentences, more often than not, the inside-outside algorithm converges to a suboptimal grammar. At its start (after initialization from a uniform distribution over rules, and a distribution over words as described above), the inside-outside algorithm is "guided by tree-counting arguments" (de Marcken, 1995) since the estimated probability of a rule is proportional to the number of parse trees the rule participates in and "not mutual information between words [in the training corpus]" (de Marcken, 1995) since the grammar is unlexicalized. In subsequent iterations, the algorithm is "incapable of performing a real search over the parameter space. Instead it converges on the nearest grammar in which the terminals concentrate probability on a small number of rules" (de Marcken, 1995). The experiments illustrates the sensitivity of the algorithm to the form of the grammar, and to how many different analyses it permits of a sentence. De Marcken's proposal is that a different conception of phrase structure grammar is required in order for inside-outside to successfully induce grammars. He suggests that the search space be simplified by flattening out phrase structure rules; he proposes using *Link Grammars* that are represented in terms of head relations. Link grammars factor out many details of phrase structure grammar that make the search space complex. de Marcken (1995)'s experiments with dummy corpora and a bare phrase structure grammar with no word-level dependencies demonstrates that a symbolic backbone with good properties is essential for inducing useful grammars.

Pereira and Schabes (1992)

In order to get around the problem that PCFGs induced by inside-outside do not necessarily yield tree structures that are desired as the output of parsing, Pereira & Schabes (1992) modified the inside-outside algorithm to take advantage of a training corpus with partial or complete bracketing information. The new algorithm has better time-complexity and convergence properties. It has the added advantage that the estimated models result in structures that are similar to the bracketing structures provided in the training corpus. Thus, the new algorithm learns grammars that are interpretable according to linguistic notions. In the main experiment in the paper, the training corpus consists of sequences of part of speech tags extracted from the Air Travel Information System (ATIS) corpus of transcribed speech, and a bracketing of these sequences derived from the parse trees for them from the relevant subset of the Penn Treebank. The modified algorithm is used to induce a probabilistic grammar model from these sequences. The modified algorithm performs much better than inside-outside on raw sequences, as evaluated on bracketing accuracy on a test data set of about 70 sequences. The two methods (standard and modified inside-outside) result in grammars that stabilize at very close values of cross-entropy, although the cross-entropy of models obtained with the modified algorithm initially reduces faster. A qualitative analysis of parses using models learnt by the two algorithms shows that the modified algorithm leads to much better linguistic structures. This result is not surprising, since the training data contains bracketing information. While partial or fully bracketed training data are much easier to obtain than fully labeled and bracketed training data, it still remains an expensive task to create large amounts of bracketed training data. Hence it is not clear to what extent the modified algorithm can be used practically to induce broad-coverage grammars.

Carroll and Rooth (1998)

Some of the more successful experiments using inside-outside to induce large-scale grammars from natural language corpora (and not toy or artificial grammars) have been performed by Carroll & Rooth (1998) for English, and Beil et al. (1999) for German. Carroll & Rooth (1998) use a hand crafted X-bar grammar which contains complementation rules for heads like verbs, nouns, adjective, and prepositions. The context-free grammar is headed and lexicalized. They use a variation of the EM algorithm, computing expectations in the standard way, but employing a maximization step that involves smoothing of lexicalized rules against unlexicalized rules using Katz's backoff scheme (Katz, 1987), in order to make the parameter space manageable. In one iteration, a training corpus of 5 million words is used, followed by the next iteration with a different 5 million word corpus. Eight such iterations are run. The estimation of the lexicalized models is bootstrapped by first using an unlexicalized model to collect lexicalized event counts. The induced complement frames of verbs are evaluated against a dictionary, with fairly good recall and precision scores. They also measured cross entropy for three test verbs, obtaining a reduction in cross-entropy in the first few iterations. Beil et al. (1999) run a similar experiment to obtain a model for German subordinate clauses. While these results are encouraging as to the efficacy of inside-outside for inducing broad-coverage grammars, the evaluations are small scale and cannot be compared to other results since the grammars involved are hand-crafted grammars. Since gold-standard test data was not available, their evaluations were on non-standard testsets.

3.5 Inside Outside Re-estimation of lexical parameters of Treebank PCFGs

In order to model complex linguistic phenomenon, it is necessary to have grammars with complex representations. Such grammars have a large number of parameters, making them harder to estimate using EM due to a corresponding increase in the number of local maxima of the objective function that is being maximized. At the same time unsupervised methods are important for these complex models, since they are also harder to estimate using annotated data (as compared to simpler models). This is because more parameters have to be estimated from the same amount of annotated data, making the effective data more sparse. Much of previous research concludes that the issues of local maxima are too severe to use EM to induce grammars of any complexity. A lexicalized PCFG is an example. A lexicalized PCFG has a very large number of parameters which makes estimation subject to the crippling problem of local maxima. In addition, complex models with a large number of parameters also make large-scale estimation computationally impractical, even given the massive increases in readily-available computational resources. Each iteration of the inside-outside algorithm on a grammar with n non-terminals may require $n^3|w|^3$ time per training sentence w of length $|w|$ (each iteration of a finite state grammar requires s^2w time per sentence w). For example, Carroll & Rooth (1998) could not carry out full EM on their lexicalized model, and had to implement a smoothing scheme that reduced the number of parameters of the model in the maximization step.

Second is the problem of the algorithm converging to different solutions depending on the initial model; as de Marcken (1995) shows, more often than not, the convergence is to a wrong model. One way to improve re-estimation is to constrain it with a sensible

prior. A linguistically sensible and accurate prior model may be obtained by using annotated data to train it, i.e. a treebank model. An initial model trained on a treebank with a high precision and recall accuracy would also set a high baseline for the re-estimated models.

3.5.1 Our solution

We work with a Penn Treebank PCFG as the initial model for inside-outside re-estimation. In Chapter 2, we described an unlexicalized PCFG with features on head categories that encode their syntactic context. This is a form of annotation where contextual information is pushed down the tree to be marked on pre-terminal categories, rather than pushing information about heads up the tree in order to condition rules on it. To a certain extent, it achieves the same effect as lexicalization, with syntactic constructs conditioned on heads. At the same time, since only pre-terminal categories and not all the productions of the PCFG are split into subcategories, contrary to a lexicalized model, the number of parameters of the PCFG model remains small enough that inside-outside re-estimation can be carried out on a reasonably large training corpus.

A treebank grammar used as the initial model has another important advantage that all re-estimated models will have the symbolic backbone consisting of rules from the treebank (or a transformation of it). This will allow parses obtained with re-estimated grammars to be interpreted in a standard way, and also to be quantitatively evaluated on gold standard data from the treebank. Previous experiments conducted with inside-outside estimation have not used treebank grammars, but have used either artificial grammars (e.g. de Marcken (1995); Pereira & Schabes (1992)) or non-standard grammars. The evaluations of estimated models have also been non-standard, and over small

testsets for which gold standard parses were obtained by hand-annotation by the researchers themselves. For example, Pereira & Schabes (1992) use only 70 sentences of part-of-speech sequences (901 part-of-speech tokens) from the Air Travel Information System (ATIS) corpus. Carroll & Rooth (1998) and Beil et al. (1999) use hand-crafted grammars and evaluate learning of subcategorization frames over a dictionary. Using a Penn Treebank grammar as the context-free backbone will help to interpret and evaluate re-estimated grammars on a gold standard test set from the Penn treebank, a method that has become a standard way of evaluating grammar models. Thus, a linguistically sensible treebank prior might help to align the twin goals of maximizing corpus probability with the linguistic goal of maximizing interpretability, with the additional advantage of allowing objective evaluation of results.

Constraining inside-outside: syntactic parameters

It is not likely that just having a good initial model will rescue inside-outside estimation. In addition to a good initial model, one also needs a way to avoid local maxima. A treebank-trained model as the initial model already sets a fairly high baseline that the re-estimated models must surpass. Some parameters of this model are accurately estimated from the treebank itself, while others are not. For example, a good model could be obtained from the treebank of the subcategorization preference of some commonly occurring verbs, while the preferences of verbs that have occurred once or twice will not be represented accurately. In general, the Zipfian nature of word distributions would indicate that parameters related to most words are badly estimated. On the other hand, in an unlexicalized PCFG, the scarcity issue might not be so severe in the case of syntactic parameters (rule frequencies), even given that a treebank like the Penn Treebank has flat rules for some categories like NP, leading to a large proliferation of right-hand-sides.

Thus, it might be beneficial to limit unsupervised learning to lexical parameters of the PCFG, while obtaining syntactic parameters solely by supervised estimation, i.e., from a treebank. In an unlexicalized PCFG like the one described in Chapter 2, it is easy to make the distinction between structural parameters (non-terminal rules) and lexical parameters (pre-terminal to terminal rules). In terms of the iterative inside-outside procedure, retaining syntactic parameter values of the initial treebank model in all iterations places a strong constraint on the re-estimated models.

Constraining inside-outside: lexical parameters

We would like to place another constraint on the re-estimated models which ensures that re-estimated lexical distributions do not deviate from the treebank distribution in certain properties; the treebank lexicon is presumably a good representation of some general properties of the language lexicon, but is deficient in representations of specific words. For example, consider past tense and past participle forms of verbs. These have a certain proportion of occurrence in the treebank lexicon: say that past tense forms of verbs are overall much more common than past participial forms. We would like to ensure that the re-estimated lexicons retain this property. Secondly, the treebank lexicon has accurate representations of high frequency words – it is to be expected that unsupervised estimation will not be as accurate as the treebank relative-frequency estimate for parameters related to these words. Therefore, at each iteration we would like to retain lexical information from the treebank model, by merging treebank lexical parameters with the re-estimated lexical parameters. We do this by linearly interpolating the re-estimated and the treebank lexicons at each iteration. These two constraints are represented in the form of a transformation T on the re-estimated models.

Figure 3.3 represents the standard iterative inside-outside procedure. In the standard

procedure, a model re-estimated in one iteration is used in the E-step of the next iteration to compute the expected complete data frequencies. A block diagram representation of the modified procedure is shown in Figure 3.4 – here, a transformation is performed on a re-estimated model before it is used to compute expectation in the next iteration. The two procedures and the transformation T is described formally in the following section.

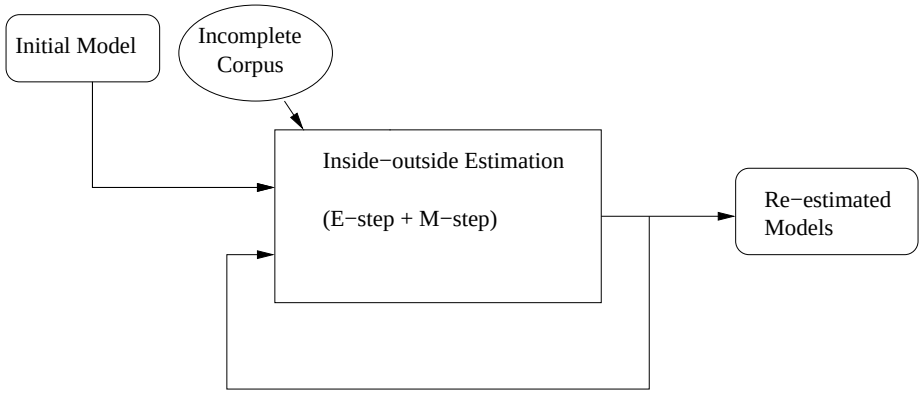


Figure 3.3: Block diagram showing standard iterative Inside-Outside estimation

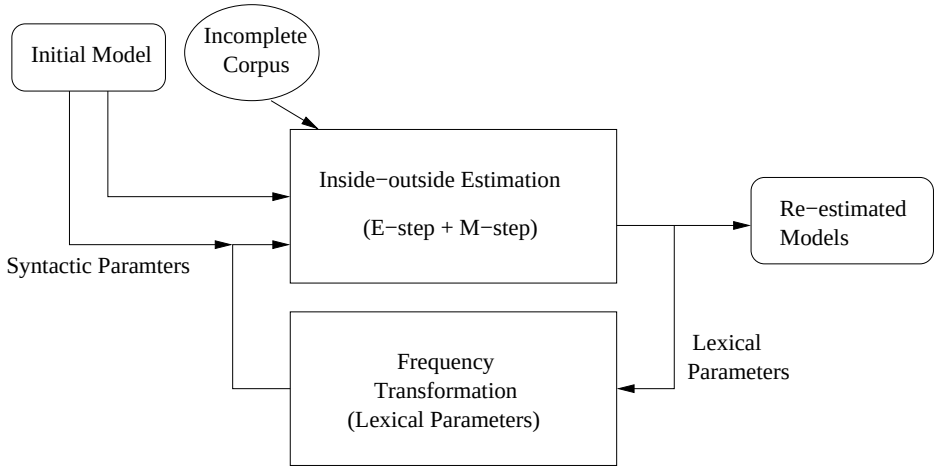


Figure 3.4: Block diagram showing interleaving of a lexical transformation between iterations of Inside-outside estimation.

3.6 Formal description of the Modified Inside-outside Procedure

We represent the standard and modified inside-outside iterative procedures formally in this section. The concepts behind maximum-likelihood estimation from incomplete data (E-step and M-step for PCFG estimation) were described in §3.2.1. In this section, our concern is the iterative procedure, and the transformation on the re-estimated models that we introduce. Hence, we do not represent the details of the E-step and M-step corresponding to each iteration – these are standard maximum-likelihood estimations over expectations computed using inside and outside probabilities.

3.6.1 Standard Inside-Outside Re-estimation

We represent the standard iterative inside-outside re-estimation procedure in the following simple form (Equation 3.2). We use the notation $I(C, e)$ to designate the new frequency model, computed via inside-outside from the corpus C by using a probability model based on the frequency model e. Each successive frequency model e_{i+1} is estimated from the corpus C using a probability model determined by the previous frequency model e_i. Our notation always refers to frequency models such as e_i, rather than the relative-frequency probability models they determine. The corpus C is the incomplete data corpus. The context-free symbolic backbone is obtained from a treebank (the transformed Penn Treebank in our case).

Our starting model is t, a smoothed frequency model over the treebank corpus. The smoothing procedure is described in detail in Chapter 4, §4.4. The purpose of the smoothing is to merge the corpus data with the treebank model – this is necessary because the corpus may contain novel words. Parameters related to these words have to be

given a non-zero value in the initial model, or else new values for them will never be induced. The smoothed treebank model is used as the initial model for the inside-outside re-estimation procedure.

$$
\begin{aligned}
e_0 &= t \\
e_1 &= I(C, e_0) \\
e_2 &= I(C, e_1) \\
&\ldots \\
e_{i+1} &= I(C, e_i) \\
&\ldots
\end{aligned}
\tag{3.2}
$$

3.6.2 Interleaved Inside-outside Re-estimation

In order to constrain re-estimation, we define a modified inside-outside procedure in which a frequency transformation $T(c, t)$ is interleaved between the iterations of the standard inside-outside procedure. The form of this interleaved procedure is shown in Equation 3.3.

$$
\begin{aligned}
d_0 &= t && \textit{smoothed treebank model} \\
c_1 &= I(C, d_0) && \textit{estimation step} \\
d_1 &= T(c_1, t) && \textit{transformation step} \\
c_2 &= I(C, d_1) && \textit{estimation step} \\
d_2 &= T(c_2, t) && \textit{transformation step} \\
&\ldots \\
c_{i+1} &= I(C, d_i) && \textit{estimation step} \\
d_{i+1} &= T(c_{i+1}, t) && \textit{transformation step} \\
&\ldots
\end{aligned}
\tag{3.3}
$$

In Equation 3.3, t designates the smoothed treebank model. For each iteration i, c_i represent the maximum-likelihood distributions obtained by inside-outside estimation over the corpus C using a model d_{i-1}. d_i represent *derived* distributions obtained by performing a transformation T on c_i. The transformation T combines the re-estimated model c_i and the smoothed treebank model t and is described below.

Notation

As before in Chapter 2, $t(w, \tau, \iota)$ represents the frequency of lexical parameters (pre-terminal to terminal rule frequencies) for the treebank model t, and $c_i(w, \tau, \iota)$ represents the frequency of lexical parameters for the re-estimated models c_i, where w is the terminal word, τ is a (Penn Treebank-style) part-of-speech tag, and ι is the sequence of additional features attached to the part-of-speech tag. Each tag-incorporation combination $\tau_a.\iota_b$ is a parameter of the PCFG and forms a partition of the context-free grammar, consisting of pre-terminal rules of the form shown below.

$$\tau_a.\iota_b \rightarrow w_1$$
$$\tau_a.\iota_b \rightarrow w_2$$
$$\tau_a.\iota_b \rightarrow w_3$$
$$\ldots$$
$$\tau_a.\iota_b \rightarrow w_n$$

A marginal frequency is defined by summation, for tag-incorporation sequences in Equation 3.4 and for tags in Equation 3.5.

$$f(\tau, \iota) = \sum_w f(w, \tau, \iota). \tag{3.4}$$

$$f(\tau) = \sum_{w} \sum_{\iota} f(w, \tau, \iota). \tag{3.5}$$

Transformation

The transformation T is used to obtain the derived models d_i and consists of two parts, corresponding to the syntactic and the lexical parameters of d_i:

Syntactic Parameters

The syntactic parameters of d_i are copied from t. These are simply rule frequencies obtained from the (transformed) treebank corpus. This step is like resetting the re-estimated rule frequencies to the original values in the initial treebank model. The motivation behind this step has already been discussed in §3.5.1.

Lexical Parameters

In the case of lexical parameters, we would like the re-estimation models to retain new parameter values from the maximum-likelihood estimate over the training corpus. At the same time, we would like to constrain the distribution so that it does not drift away from the treebank lexical model and retains similarity to it in some aspects . We therefore merge the two models by linearly interpolating them and thus obtain the lexical parameters of the derived model d_i.

To obtain the lexical parameters of d_i:

- First, the frequencies in the re-estimated (corpus) model $c_i(w, \tau, \iota)$ are scaled by

the ratio of treebank and corpus marginal frequencies, as shown in Equation 3.6.

$$\bar{c_i}(w, \tau, \iota) = \frac{t(\tau, \iota)}{c_i(\tau, \iota)} c_i(w, \tau, \iota). \tag{3.6}$$

- Second, lexical parameters from the treebank model *t* and lexical parameters from the re-estimated model are linearly combined, shown in Eq. 3.7.

$$d_i(w, \tau, \iota) = (1 - \lambda_{\tau,\iota}) t(w, \tau, \iota) + \lambda_{\tau,\iota} \bar{c_i}(w, \tau, \iota) \tag{3.7}$$

where $\lambda_{\tau,\iota}$ is a parameter with $0 < \lambda_{\tau,\iota} < 1$ which may depend on the tag and incorporation.

Marginal Frequencies

Since we represent the treebank and re-estimated models as frequency rather than relative-frequency models, it is important to ensure that the marginal frequencies of tag-incorporation combinations in the derived models are the same as the treebank marginals. This ensures that the derived models d_i are still treebank PCFGs, as defined in §3.2.2.

Equations 3.8 and 3.9 shows that the tag-incorporation marginal frequency of the derived model equals the treebank model.

$$\begin{aligned}
\bar{c}(\tau, \iota) &= \textstyle\sum_w \bar{c}(w, \tau, \iota) \\
&= \textstyle\sum_w \frac{t(\tau,\iota)}{c(\tau,\iota)} c(w, \tau, \iota) && \text{– From Eq. 3.6} \\
&= \textstyle\frac{t(\tau,\iota)}{c(\tau,\iota)} \sum_w c(w, \tau, \iota) \\
&= \textstyle\frac{t(\tau,\iota)}{c(\tau,i)} c(\tau, \iota) \\
&= t(\tau, \iota)
\end{aligned} \tag{3.8}$$

$$
\begin{aligned}
d(\tau, \iota) &= \textstyle\sum_w d(w, \tau, \iota) \\
&= \textstyle\sum_w (1 - \lambda_{\tau,\iota}) t(w, \tau, \iota) + \lambda_{\tau,\iota} \overline{c}(w, \tau, \iota) \\
&= (1 - \lambda_{\tau,\iota}) \textstyle\sum_w t(w, \tau, \iota) \\
&+ \ \lambda_{\tau,\iota} \textstyle\sum_w \overline{c}(w, \tau, \iota) \\
&= (1 - \lambda_{\tau,\iota}) t(\tau, \iota) + \lambda_{\tau,\iota} \overline{c}(\tau, \iota) \\
&= (1 - \lambda_{\tau,\iota}) t(\tau, \iota) + \lambda_{\tau,\iota} t(\tau, \iota) \qquad \text{– From Eq. 3.8} \\
&= t(\tau, \iota)
\end{aligned}
\tag{3.9}
$$

According to our definition of treebank PCFGs in §3.2.2,

$$\sum_{r \in G_A} p(r) = 1$$

$$p(r) = \frac{f(r)}{\sum_{r \in G_A} f(r)}$$

where G_A is a partition of the CFG with left-hand-side A.

For the treebank model t,

$$f(r) = t(w, \tau, \iota)$$

This is because each terminal rule r of the treebank PCFG is of the form $\tau.\iota \rightarrow w$. Thus, each symbol $\tau.\iota$ forms a partition G_A where $A = \tau.\iota$. The treebank frequency of each word w occurring with a particular τ and ι is $t(w, \tau, \iota)$ and thus corresponds to the frequency of the rule r.

Thus,

$$\sum_{r \in G_A} f(r) = \sum_w t(w, \tau, \iota)$$

$$= t(\tau, \iota)$$

$$p(r) = p(\tau.\iota \rightarrow w) = \frac{t(w, \tau, \iota)}{t(\tau, \iota)}$$

For the derived model,

$$f(r) = d(w, \tau, \iota)$$

$$\begin{aligned} \sum_{r \in G_A} f(r) &= \sum_w d(w, \tau, \iota) \\ &= t(\tau, \iota) \qquad \text{–From Eq. 3.9} \end{aligned} \tag{3.10}$$

$$p(r) = \frac{d(w, \tau, \iota)}{t(\tau, \iota)}$$

$$\begin{aligned} \sum_{r \in G_A} p(r) &= \sum_{r \in G_A} \tfrac{d(w,\tau,\iota)}{t(\tau,\iota)} \\ &= \tfrac{\sum_{r \in G_A} d(w,\tau,\iota)}{t(\tau,\iota)} \\ &= \tfrac{t(\tau,\iota)}{t(\tau,\iota)} \\ &= 1 \end{aligned} \tag{3.11}$$

Thus, in the derived distribution, the total frequency allocated to G_A remains the same as the treebank model. This frequency is distributed amongst the right-hand-sides of G_A. Since syntactic parameters in d are identical to the treebank model t, Equation 3.11 means that d is a probability distribution, as defined in §3.2.2.

Effect of the transformation

The lexical transformation can be interpreted in the following way:
The scaling of the corpus relative frequencies in Equation 3.6 does not have an effect on the probability model determined by the derived frequency model d. This is because in order to obtain the relative frequency model corresponding to d, it has to be normalized by the marginal $d(\tau, \iota)$. Thus the normalized form of Equation 3.7 is

$$\frac{d(w,\tau,\iota)}{d(\tau,\iota)} = (1-\lambda_{\tau,\iota})\frac{t(w,\tau,\iota)}{d(\tau,\iota)} + \lambda_{\tau,\iota}\frac{\bar{c_i}(w,\tau,\iota)}{d(\tau,\iota)} \tag{3.12}$$

Substituting from Equations 3.9 and 3.6,

$$\frac{d(w,\tau,\iota)}{d(\tau,\iota)} = (1-\lambda_{\tau,\iota})\frac{t(w,\tau,\iota)}{t(\tau,\iota)} + \lambda_{\tau,\iota}\frac{t(\tau,\iota)}{c(\tau,\iota)}\frac{c_i(w,\tau,\iota)}{t(\tau,\iota)} \tag{3.13}$$

$$\frac{d(w,\tau,\iota)}{d(\tau,\iota)} = (1-\lambda_{\tau,\iota})\frac{t(w,\tau,\iota)}{t(\tau,\iota)} + \lambda_{\tau,\iota}\frac{c_i(w,\tau,\iota)}{c(\tau,\iota)} \tag{3.14}$$

However, scaling the corpus frequency model by the ratio of treebank to corpus marginal frequencies lets us interpret the derived model as taking a certain frequency away from the treebank model and distributing it to the re-estimated model in proportion to the re-estimated relative frequencies.

Example illustrating interpolation of treebank and re-estimated lexicons

In order to illustrate the effect of the interpolation, we present an example with dummy lexicons. Let *t* be the treebank lexicon and *c* be the re-estimated lexicon. Each contain two words A and B. Let these words be verbs, so that the tags associated with them are verbal tags (like VBD and VBN). In practice, the transform in Equation 3.7 acts on all words in the lexicon, not just verbs.

In the first example (Figure 3.5), we consider a treebank lexicon with A and B occurring with just one tag and incorporation combination “VBD.x”. In the corpus model as well, these words occur with just that combination. However, in the corpus the word B is more common than word A. This might occur (for instance) if the corpus is from a different domain from the treebank. Derived models *d*, obtained by using different values of the parameter λ are shown. A value of $\lambda = 0.5$ gives equal weight to the treebank

and the scaled corpus model. A values of $\lambda = 0.1$ gives more weight to the treebank model, while a values of $\lambda = 0.9$ gives more weight to the scaled corpus model. The total treebank frequency of VBD.x (= 110) is split between A and B in different proportions. The column labeled d_1 shows the values in the derived model when the treebank and corpus models are given equal weight. In column d_2, the treebank model is given more weight ($\lambda = 0.1$) and the derived model is closer to the treebank model. When the value of λ is high (as in column d_3 with $\lambda = 0.9$), the proportions of A and B in the derived model are much more similar to their proportions in the corpus model.

	t	c	d_1, $\lambda = 0.5$	d_2, $\lambda = 0.1$	d_3, $\lambda = 0.9$
A	VBD.x 100	VBD.x 1000	VBD.x 59.16	VBD.x 91.83	VBD.x 26.5
B	VBD.x 10	VBD.x 5000	VBD.x 50.83	VBD.x 18.16	VBD.x 83.5

Figure 3.5: Example illustrating interpolation of treebank and re-estimated lexicons

In the second example (Figure 3.6), each word occurs with two tags VBD and VBN. Again, in the derived model, the total treebank frequency for VBD.x (= 110) is split between word A (A.VBD.x) and word B (B.VBD.x). The treebank frequency of 9 for the second tag-incorporation combination VBN.x is similarly split between A.VBN.x and B.VBN.x. Notice that when $\lambda = 0.1$, the proportion of the total frequency given to the words A and B is closer to the treebank model, but is much closer to their corpus proportion when $\lambda = 0.9$. In both cases however, the proportions between the tags VBD and VBN remains identical in the two lexicons d_2 and d_3 (as also in d_1) and equal to the treebank proportion of the two tags.

In the third example (Figure 3.7), each tag for A and B has two incorporations x and y. Word A is more common than word B in the treebank lexicon, but B is much more frequent in the corpus lexicon. The frequencies are stipulated so that for word A,

	t	c	d_1, $\lambda = 0.5$	d_2, $\lambda = 0.1$	d_3, $\lambda = 0.9$
A	VBD.x 100	VBD.x 1000	VBD.x 59.16	VBD.x 91	VBD.x 26.5
	VBN.x 5	VBN.x 50	VBN.x 3.4	VBN.x 4.68	VBN.x 2.12
B	VBD.x 10	VBD.x 5000	VBD.x 50.83	VBD.x 18.16	VBD.x 83.5
	VBN.x 4	VBN. x 200	VBN.x 3.25	VBN.x 4.32	VBN.x 6.88

Figure 3.6: Example illustrating interpolation of treebank and re-estimated lexicons

the incorporation y is more frequent than x for the tag VBD in the treebank model. On the other hand, for word B, the incorporation x is more frequent than y in the treebank model. For the tag VBN, there are not enough occurrences in the treebank for there to be an accurate distribution, which is frequently the case (word A occurs with VBN only 5 times). In the corpus model, for word A, x has more frequency than y, unlike the treebank model. For word B in the corpus model, the proportions of x and y are similar to those for the word B in the treebank model.

The columns d_1, d_2 and d_3 show the derived distribution for three values of the interpolation parameter λ. For all derived models, the total frequency of the two tags VBD and VBN remain in the same proportion to each other as in the treebank model. What changes with the value of λ is the proportion of total frequency allocated to each word, and the proportion allocated to each incorporation of a tag of a word. Thus the proportion of total frequency allocated to words A and B in $d2$ (when $\lambda = 0.1$) is much closer to the treebank proportion of these words (A is more frequent than B), while it is closer to their corpus proportions when $\lambda = 0.9$ (B is more frequent than A). As as example of frequencies allocated to incorporations of a tag for words, note: when $\lambda = 0.1$, the proportion of frequency of A.VBD.x to B.VBD. x is much closer to their treebank proportions than to their corpus proportions. When $\lambda = 0.9$, their proportion is much

more similar to their corpus proportion. Thus when $\lambda = 0.1$, A.VBD.x is slightly more frequent than B.VBD.x, like in the treebank model. However, when $\lambda = 0.9$, A.VBD.x is much less frequent than B.VBD.x, like in the corpus model.

Thus, the interpolation imposes general constraints on the nature of the lexicon.

	t	c	d_1, $\lambda = 0.5$	d_2, $\lambda = 0.1$	d_3, $\lambda = 0.9$
A	VBD.x 10	VBD.x 900	VBD.x 6.65	VBD.x 9.33	VBD.x 3.97
	VBD.y 90	VBD.y 100	VBD.y 49.18	VBD.y 81.83	VBD.y 16.52
	VBN.x 2	VBN.x 40	VBN.x 1.57	VBN.x 1.91	VBN.x 1.22
	VBN.y 3	VBN.y 10	VBN.y 1.72	VBN.y 2.74	VBN.y 0.7
B	VBD.x 8	VBD.x 4000	VBD.x 11.34	VBD.x 8.66	VBD.x 14.02
	VBD.y 2	VBD.y 1000	VBD.y 42.81	VBD.y 1.16	VBD.y 75.47
	VBN.x 2	VBN. x 100	VBN.x 2.42	VBN.x 2.08	VBN.x 2.77
	VBN.y 2	VBN.y 100	VBN.y 3.27	VBN.y 2.25	VBN.y 4.29

Figure 3.7: Example illustrating interpolation of treebank and re-estimated lexicons

3.7 Chapter Summary

In this chapter, we presented the idea of re-estimating the lexical parameters of a treebank PCFG. The idea ties into several aspects of estimation of PCFGs. Firstly, the problem of lexical scarcity in treebank PCFGs is more severe than that of syntactic scarcity, hence there is a need to use unsupervised techniques. The treebank PCFG that we design is unlexicalised but with complex lexical categories – having complex categories increases the severity of the lexical scarcity problem. However, using an unlexicalised

PCFG allows us, firstly, to use a large corpus for estimation due to the small size of grammar, and secondly, to retain syntactic parameters from the treebank model without re-estimation, since they are not as sparse as lexical parameters. This imposes a strong constraint on the re-estimated models. At the same time, the treebank PCFG provides a good initial model for estimation and allows objective evaluation of re-estimated grammars against an established gold standard. We also presented the idea of smoothing re-estimated lexicons with the treebank lexical model, as a way to ensure that re-estimated models retain some properties of the treebank model. Thus, we constrain re-estimation by using treebank PCFG relative-frequency estimates, rather than heuristic constraints. In the next chapter, we present re-estimation experiments with a treebank PCFG, and empirical evaluations and comparisons of the standard and modified inside-outside procedure presented in this chapter.

CHAPTER 4

INSIDE-OUTSIDE RE-ESTIMATION: EXPERIMENTS AND RESULTS

4.1 Introduction

The previous chapter presented the theoretical idea and motivation for the re-estimation of lexical parameters of a treebank PCFG from unannotated data. We introduced a procedure whereby lexical transformations are interleaved between iterations of inside-outside. This chapter describes re-estimation experiments and presents results and analyses of re-estimated models. The goal of the experiments is an empirical evaluation of the efficacy of inside-outside estimation for learning better lexical parameters for PCFGs, and a comparison of models obtained with the interleaved procedure with models obtained with standard inside-outside re-estimation.

We perform standard inside-outside estimation with 4 million words of unannotated Wall Street Journal text and interleaved estimation with 4, 8 and 12 million words (Deoskar, 2008). The re-estimated models are evaluated on two counts: (i) labeled bracketing score of Viterbi parses of sentences in Penn Treebank section 23 obtained with the re-estimated models, and (ii) identification of correct subcategorization frames for verbs in Viterbi parses of test sentences. We present a method of smoothing the treebank model to obtain an initial model for inside-outside estimation which has good properties in order to induce parameters related to new words. We perform several analyses of the re-estimated models, to better understand the learning of different types of subcategorization frames. We also analyze our results in order to understand the relative efficacy of treebank data versus unlabeled data for learning parameters of words of different occurrence frequencies in the labeled and unlabeled corpora.

In the following sections, we first describe the experimental setup (§4.2), followed by a description of our evaluation measures (§4.3), and the procedure for obtaining the initial model for re-estimation from the treebank PCFG (§4.4). The rest of the chapter contains various re-estimation experiments, followed by evaluations of re-estimated models and their analyses.

4.2 Basic Experimental Setup

Inside-outside is an estimation procedure that, given an initial model and a corpus of unannotated data, gives maximum-likelihood models that maximize corpus-likelihood at each iteration. Multiple iterations are required for convergence. We conduct re-estimation experiments over a parallel computing cluster given an initial treebank model and a corpus C . The experimental procedure is represented as a block diagram in Figure 4.1. First, the unannotated training corpus C is POS-tagged with a tagger (Treetagger, (Schmid, 1994)). We obtain a model t by merging a treebank-trained PCFG with the POS-tagged corpus C_{pos}: this model t is used as the initial model for inside-outside re-estimation. Since the training corpus is quite large (a minimum of 4 million words), we carry out re-estimation on portions of the corpus in parallel. Therefore, the corpus C is split into n parts, $C_1 ... C_n$ at sentence boundaries. Inside-outside re-estimation using the model t and the sub-corpus C_i is carried out in parallel on n separate processors, resulting in n re-estimated frequency models. The n re-estimated models are then merged together by adding frequencies, after removing zero frequency items from each sub-model (each sub-model contains a large number of zeroes, especially for lexical parameters, since it is estimated using only a part of the training corpus)[1]. A lexical transformation may be

[1]This is only an implementation detail– removing zeroes does not affect the mathematical properties of the model. However, it substantially reduces the size of the models, making it possible to use Lopar (Schmid, 2000a) for merging.

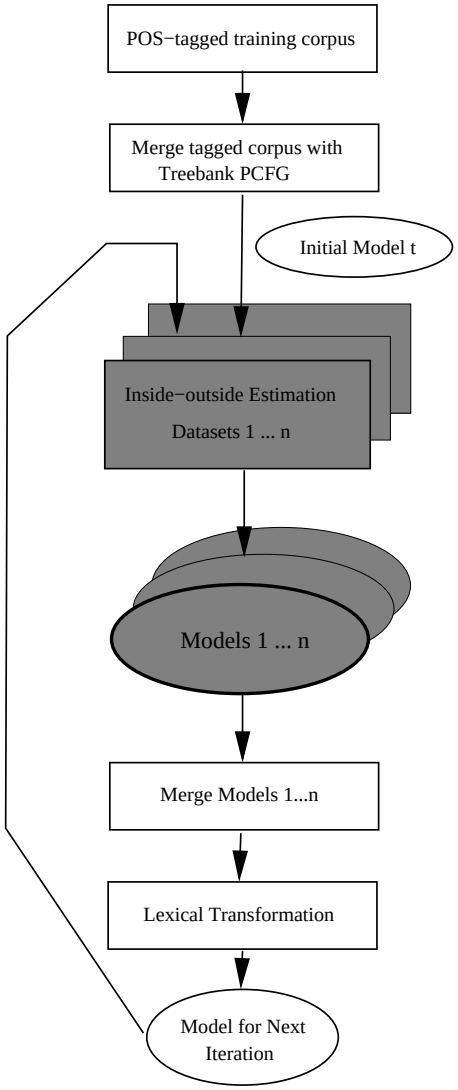


Figure 4.1: Flow chart showing experimental procedure for Inside-outside Re-estimation

carried out on the merged model, which results in a new model (grammar and lexicon.) This model is used in the next iteration of inside-outside. The experimental setup of re-estimating separate frequency models for each sub-corpus in parallel and merging the resultant models makes it easy to scale up the size of training data on a parallel computing cluster.

The following is the list of components required for re-estimation experiments.

1. A treebank-trained PCFG t_0, containing features related to structure-selection incorporated into pre-terminal categories.
2. A training data set C_{pos}, tagged with part-of-speech tags. We use Treetagger (Schmid, 1994) for POS tagging.
3. A smoothed treebank model t with the training data C_{pos} merged in. t is used as the initial model for inside-outside re-estimation.
4. The software Bitpar (Schmid, 2004) for inside-outside estimation.
5. Software for lexical transformation.
6. Software for merging models re-estimated in parallel, over separate sub-corpora obtained by splitting the training corpus C. We currently use the parser Lopar, (Schmid, 2000a) which has an option for merging frequency models.
7. A computing cluster with appropriate number of nodes for re-estimation.

4.3 Evaluations

The goal of our experiments is to learn lexically specific information such as the structure-selection properties of words. Verbal subcategorization is the most important example of structure-selection by a head word. Hence we focus on evaluating re-estimated models on measures related to the subcategorization frames of verbs (and also nouns, to a lesser extent). Secondly, as a coarse measure, we report labeled bracketing precision, recall and *f*-score of Viterbi parses of test sentences. Since the re-estimated models are treebank PCFGs, we present labeled bracketing evaluations of Viterbi parses of the standard test section 23 of the Penn Treebank. The following sections describe each evaluation measure in detail.

4.3.1 Subcategorization evaluation

We focus on learning verbal subcategorization as a typical case of lexico-syntactic information. Probabilistic subcategorization information for verbs is known to be useful to find the correct parse. For example, Collins's (1997) Model II improved performance significantly when a distinction was made between arguments and adjuncts in the model. To improve their parser's PP attachment performance, Basili et al. (1997) use a lexicon of automatically acquired subcategorization frames. Carroll et al. (1998) obtain an improvement in parser performance by using a subcategorization lexicon acquired automatically by using an unlexicalized statistical parser.

The subcategorization frame (SF) of verbs is a parameter of our PCFG – verbal tags in the PCFG are followed by a feature sequence that denotes the subcategorization frame for that verb. We evaluate the re-estimated models on the task of detecting correct frames for verbs in maximum-probability (Viterbi) parses obtained using the models. Since the sequence of features incorporated on a verb (i.e., its subcategorization frame) correlates with a particular tree-structure (local and non-local) associated with the verb, a correct subcategorization frame indicates a correct structure in the Viterbi tree.

The evaluation of re-estimated models on the basis on maximum probability parses has two aspects to it. On the one hand, it allows us to evaluate the models in a "real" setting. If subcategorization probabilities are meant to improve parsing, then our lexicons must be evaluated on token accuracy in a parsing task. But on the other hand, a parsing task is an indirect measure, since in a parsing task words are not considered in isolation, and global optimization constraints in a sentence might override local ones[2].

[2]It is for this reason that measures like cross-entropy are sometimes used for evaluation of lexicons. For instance, Eisner (2001) evaluates his transformational lexicons using the measure of RHS perplexity, or conditional cross-entropy, following Carroll & Rooth (1998).

In the parsing-based subcategorization evaluation, all tokens of verbs and their pre-terminal symbols (consisting of a POS tag and an incorporation sequence encoding the SF) are extracted from the Viterbi parses of sentences in a test set. This tag-SF sequence is compared to corresponding features on that verb in the feature-structure tree for that sentence, which forms our gold standard for comparison. The token is scored correct if the two match exactly. POS errors are scored as incorrect, even if the SF is correct. Since we do not POS tag a test sentence before passing it to the parser– the POS-tag- SF combination for a verb in the Viterbi parse is the pre-terminal category assigned to that word by the parser. The gold standard is obtained from feature-structure trees, which are obtainable for each Penn Treebank tree, as described in Chapter 2. The feature-structure tree contains all features in the feature-constraint grammar on non-terminal nodes, out of which a subset is incorporated into the PCFG category symbols in an ordered sequence. This subset of features and their values are extracted from the feature-structure tree to obtain a comparable gold standard tree to the Viterbi tree.

Subcategorization frames in the grammar

In the version of PCFGs used in the experiments in most of this chapter (version b14), three features are incorporated into verbal categories. The first feature is `Val` (for valence) and denotes basic categories of subcategorization such as transitive, intransitive, ditransitive, NP-PP, S, etc. The values of the `Val` feature corresponds to the shape of RHS of the VP rule, i.e. the sisters of the verbal pre-terminal category that are complements of the verb. The `Val` feature was described in Chapter 2 §2.4.2. The entire list of values of `Val` and the corresponding rule shape is given in Chapter 2, Table 2.1. The second feature `Vsel` denotes, for clausal complements, the type of clause (finite, infinite, small clause, etc.). The third feature encodes the nature of the subject of the clausal comple-

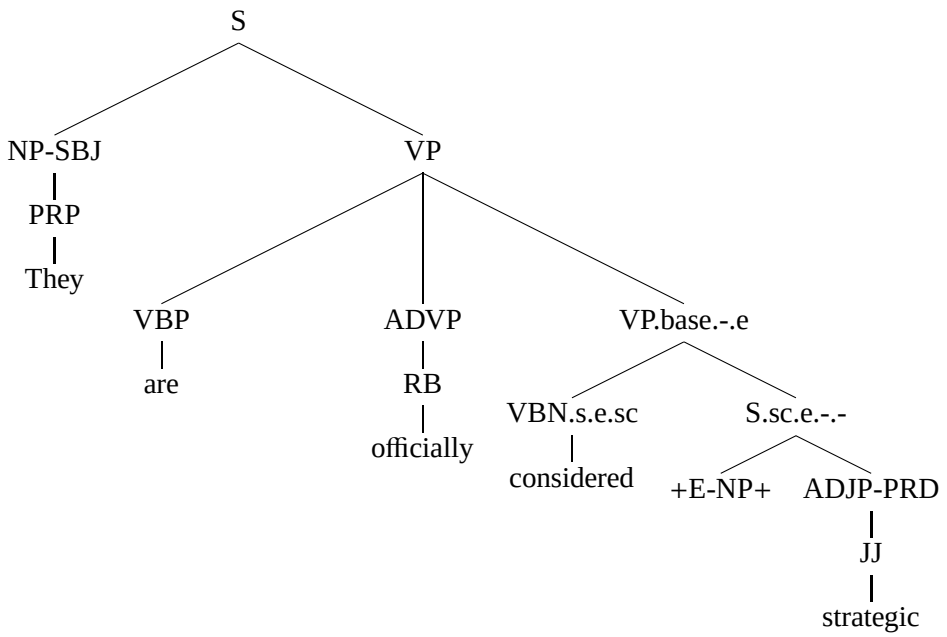


Figure 4.2: A verb *considered* with a small clause subcategorization frame

ments (empty category or non-empty). These three features together comprise what is called the subcategorization frame for the verb. See §2.5.9 for a list of all rules on verbal categories.

As an example, consider the treebank sentence *They are officially considered strategic*. The Penn Treebank tree for this sentence, along with our feature incorporations on the pre-terminal node for the verb *considered* is shown in Figure 4.2. Features on higher nodes in the tree are omitted for clarity. In Penn Treebank annotation, the phrase *considered strategic* is analyzed as a VP. The verb *considered* has a small clause complement, indicated by an S does that does not dominate a VP. The small clause in this example has an empty subject +E-NP+, which is the trace of a passive. In a correct parse of this sentence, the verb *considered* will get a pre-terminal sequence of VBN.s.e.sc. This sequence indicates a passive participle verb (VBN) with a clausal complement (s) which is of the type *small clause* (sc), and has an empty subject (e). Thus, getting this subcatego-

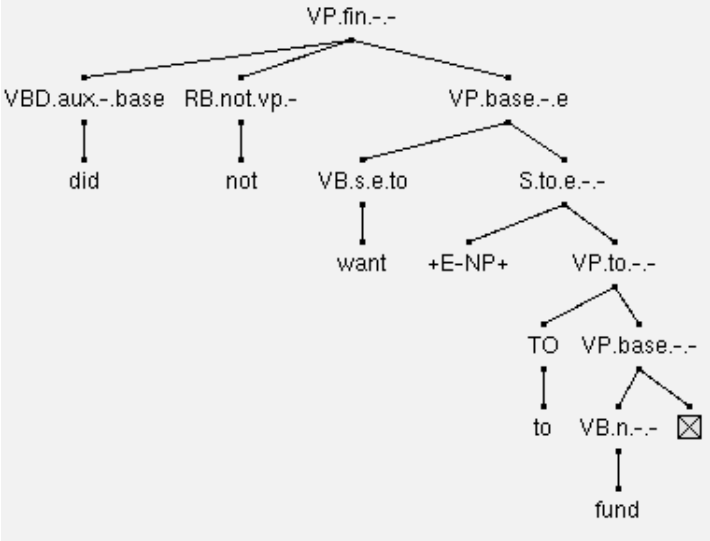


Figure 4.3: Example subcategorization frame for a control verb *want*. The sequence of features incorporated on the verbal POS tag are (left-to-right) Val (valence), Sbj (subject) and Vsel.

rization frame correct in a Viterbi parse implies correctly identifying the following: the past participle verb, its clausal complement (S), the structure of the clausal complement (an S without a VP, i.e., a *small clause*), and the empty subject of type +E-NP+ of the clausal complement.

Similarly, a control verb like *want* in the sentence *did not want to fund...* gets a pre-terminal sequence of VB.s.e.to, since the clausal complement in this case is an infinitival clause. Figure 4.3 shows the subcategorization frame on the verb *want* in such a configuration. This figure also shows all the feature incorporations on all categories in the tree. The subcategorization frame VB.s.e.to in this case implies a clausal (S) complement, with an empty subject, and a infinitival VP headed by TO. Thus, identifying this frame correctly in a Viterbi parse implies getting all of this structure right.

We have a total of 81 categories of SFs, without counting specific prepositions in prepositional frames, making fairly fine-grained distinctions of verbal categories. In

Table 4.1: Some verbal subcategorization frames and their interpretation: a full list along with their relative frequencies and example sentences for each is given in Appendix B.

z.-.-.-	intransitive
n.-.-.-	transitive (NP)
p.-.-.-	prepositional (PP-CLR)
np.-.-.-	NP PP-CLR
s.-.-.-	S
b.-.-.-	SBAR
t.-.-.-	predicate complement
s.e.to	control
s.-.sc	active small clause complement
s.e.sc	passive small clause complement

some experiments, another version of grammar (b15) is used in which a fourth feature that encodes specific prepositions is also present on verbal categories, and is therefore a part of the subcategorization frame– this feature will take a non-default value when the feature `Val` indicates a frame that includes a PP. The prepositional head of the PP will then be a part of the subcategorization frame. Table 4.1 gives a list of some of the possible subcategorization values that verbs can take, along with the corresponding frame in our lexicon. Appendix B gives a full list of subcategorization frames, along with frequencies in our testset and some examples of each frame from the treebank.

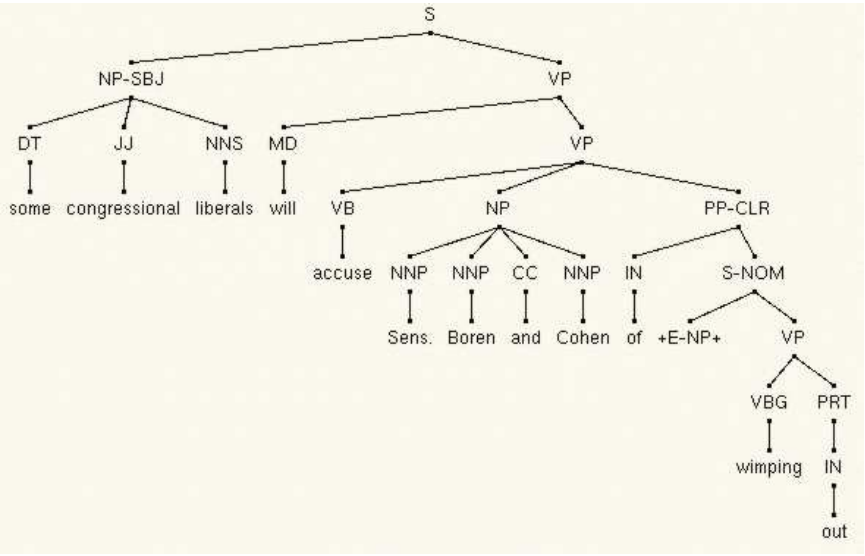
4.3.2 Labeled Bracketing evaluation

As a basic evaluation of the re-estimated grammars, we report the labeled bracketing scores on the standard test section 23 of the Penn treebank. First, the treebank anno-

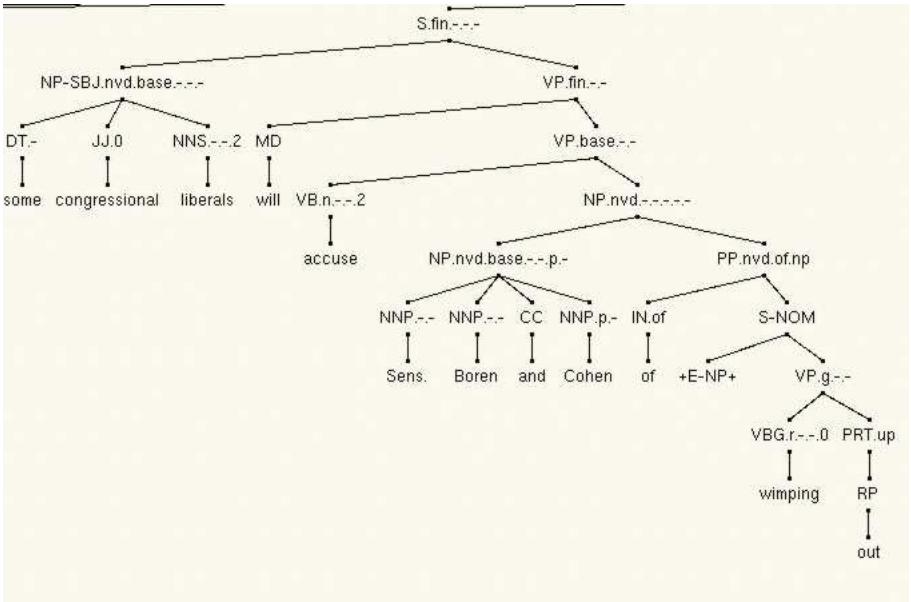
tation is stripped away from all trees in section 23, including pre-terminal tags. Using the re-estimated models, maximum probability (Viterbi) parses are obtained for all sentences. In order to accommodate unknown words from section 23, each model is first smoothed as described in Chapter 2 Equation 2.3, with the test words from section 23 forming the *corpus C*. The value of λ is 0.1, i.e., the relative weight given to the test corpus while merging it with the treebank PCFG is one tenth of the treebank model. Labeled bracketing scores are measured using the `evalb` program available from *http://nlp.cs.nyu.edu/evalb/*. Certain non-terminal symbols such as PRT and COL (which are standardly ignored by evaluations reported in the field) are not considered in the labeled bracketing evaluation.

Parseval measures and subcategorization

Labeled bracketing might be too coarse an evaluation metric to measure subcategorization acquisition. For example, Briscoe & Carroll (1997) show that subcategorization probabilities can help parsing; however they conclude that PARSEVAL measures are relatively insensitive to the argument-adjunct and attachment distinctions. The example in Figure 4.4 shows a verb *accuse* which has a PP (*of wimping out*) which is an argument of the verb *accuse* (this is seen in the gold standard treebank tree in (a)). A Viterbi parse with the PP attached incorrectly as the sister of NP is shown in Figure 4.4 (b). This incorrect attachment is reflected in the subcategorization frame on the verb *accuse* (VB.n.-.-). This incorrect attachment will be penalized in out subcategorization evaluation, since the frame will be counted as incorrect. The corresponding PARSEVAL score of the Viterbi parse shown in Figure 4.4 has a recall of 100%, a precision score of 90.4%, f-score of 94.74%, and zero average crossing, a relatively low penalty. In the case where the PP was attached correctly as daughter of the VP, but not marked as a PP-CLR (i.e.



(a) Treebank tree



(b) Viterbi tree

Figure 4.4: An incorrect PP attachment in a Viterbi parse: the correct attachment of the PP *of wimping out* is as a daughter of the VP, as shown in (a), and not as a sister of NP , as in (b).

an argument), it would not be penalized at all by the PARSEVAL metric (since the metric ignores functional marking on all categories), but would be counted as incorrect in our subcategorization metric.

4.4 Initial model for re-estimation

In order to use the treebank PCFG as an initial model for unsupervised estimation, unseen words from the unannotated training corpus must be included. If this is not done, the unsmoothed treebank model (t_0) will have zero frequency for some lexical parameters (pre-terminal to terminal rules), for example, for parameters associated with words in the unannotated corpus that were unseen in the treebank. If the probability associated with a rule is zero, all trees in the complete corpus containing that rule will have zero probability and the expectation of that rule will be zero. The inside-outside estimate $I(C, t_0)$ for the parameter would also be zero, and new lexical entries would never be induced.

Since the treebank model contains no information regarding correct incorporated feature sequences (also referred to as just *incorporations*) for unseen words, we need to assign an unseen word all incorporated feature sequences that have occurred in the treebank model for the POS tag of the word. We need to assign all possible feature sequences to *seen* words as well; although the word is seen, the correct feature sequence appropriate for a structure in a training sentence might still be unseen with that word. For seen words, the treebank probability distribution over seen feature sequences must be largely maintained, but a small frequency assigned to unseen sequences.

Thus, our smoothing scheme allocates frequency to

- combinations of words w and POS tags τ which are not present in the treebank model but are present in the corpus.
- to all possible incorporations of a POS tag for words that are present in the treebank and the corpus.

The smoothed treebank model t is obtained from the un-smoothed model t_0 as follows. First a part-of-speech tagger (Treetagger, (Schmid, 1994)) is run on the unannotated corpus C. The tagger assigns Penn Treebank style POS tags to the corpus. Tokens of words and POS tags are then tabulated to obtain a frequency table $g(w, \tau)$. Each frequency $g(w, \tau)$ is split among possible incorporations ι in proportion to a ratio of marginal frequencies in t_0:

$$g(w, \tau, \iota) = \frac{t_0(\tau, \iota)}{t_0(\tau)} g(w, \tau) \tag{4.1}$$

The smoothed model t is defined as an interpolation of g and t_0 for lexical parameters as shown in Equation 4.2, with syntactic parameters copied from t_0 (i.e., syntactic parameters are unsmoothed).

$$t(w, \tau, \iota) = (1 - \lambda_{\tau,\iota}) t_0(w, \tau, \iota) + \lambda_{\tau,\iota} g(w, \tau, \iota) \tag{4.2}$$

The value of λ is set to 0.001 for all τ and ι for all of the following experiments. Note that this smoothing procedure is the same as the procedure that is used for smoothing the treebank models before parsing test data (described in Chapter 2, §2.9). In summary, for both cases (i.e while merging a test data corpus which is to be parsed, and while merging unannotated training corpus), all feature-sequences for a POS tag are added to words from the corpus. For seen words, the treebank distribution over seen feature-sequences is largely maintained. For unseen words, an average distribution over all feature-sequences for a POS tag is added in the smoothed model. We use different values of the parameter λ while merging test data and while merging unannotated training data.

Example to illustrate properties of the initial model

We provide a small illustration to demonstrate the effect of the smoothing procedure in Equation 4.2 on the treebank lexicon.

Let t_0 represent a treebank lexicon. It contains three words A, B, and C, with part-of-speech tags T_1 and T_2 and incorporation sequences represented by small letters x, y, z, etc. The lexicon is shown in Figure 4.4, under the column labeled t_0. The treebank lexicon demonstrates the sparseness of the lexical entries. Not all incorporations or tags possible for a word are seen in the lexical entry for that word. For instance, the entry for word A does not have the tag T_1 with the incorporations y and z, which are seen with T_1 in the entry for word C.

The third column in Figure 4.4 represents frequencies $c(w, \tau)$ in a tagged corpus. These frequencies are typically larger than the treebank ones. The corpus contains the three words A, B and C that have occurred in the treebank and two novel words P and Q.

Figure 4.6 shows the model d_1 obtained by merging the treebank lexicon and the corpus, with a very low value of λ. The entries for novel verbs P and Q contain all possible incorporations for the two tags T_1 and T_2 with which they have been seen in the treebank lexicon. The entries in the merged lexicon for the seen verbs A and B also contain all possible incorporations for a particular tag (T_1 has x, y and z incorporations, while T_2 occurs with x and y). Notice that for the seen verbs A and B, the treebank distribution of frequencies over seen tag-incorporation combinations is largely maintained in the merged lexicon – thus, much of the treebank frequency is retained on T_1.x and T_2.x for A with a small frequency given to the new sequences T_1.y, T_1.z, and T_2.y.

	t_0	c
A	T_1.x 2	T_1 10 T_2 5 T_3 10
	T_2.x 1	
B	T_2.y 1	T_1 15 T_2 10
C	T_1.x 3	T_1 10 T_2 10
	T_1.y 1	
	T_1.z 1	
P	-	T_1 10 T_2 5
Q	-	T_1 5 T_2 10

Figure 4.5: Example treebank lexicon and POS tagged corpus

d_1	$\lambda = 0.001$				
A	T_1.x 1.999	T_1.y 0.0002	T_1.z 0.0002	T_2.x 0.999125	T_2.y 0.000125
B	T_1.x 0.0015	T_1.y 0.0003	T_1.z 0.0003	T_2.x 0.00025	T_2.y 0.99925
C	T_1.x 2.998	T_1.y 0.9992	T_1.z 0.9992	T_2.x 0.00025	T_2.y 0.00025
P	T_1.x 0.001	T_1.y 0.0002	T_1.z 0.0002	T_2.x 0.000125	T_2.y 0.000125
Q	T_1.x 0.0005	T_1.y 0.0001	T_1.z 0.0001	T_2.x 0.00025	T_2.y 0.00025

Figure 4.6: Smoothed Treebank model with corpus merged in, as per Equation 4.2.

4.5 Description of models, corpora and test sets used in re-estimation experiments

4.5.1 Treebank PCFG

The treebank grammars used in the experiments described in this chapter are trained on Sections 0-22 of the Penn Treebank, excluding approximately 5500 sentences: ~ 1200

sentences that are held out as Testset I and ~ 4100 sentences held out as Testset II (these testsets are described later in §4.5.3). The features on pre-terminal categories that encode structures selected by lexical items of that category and hence are relevant to our experiments are listed in Table 4.2. We use two versions of treebank PCFGs in our experiments: in one version (**b14**), all the features in Table 4.2 except for the feature `Prep` on verbal and nominal categories are incorporated into the category symbols. In the second version (**b15**), the feature `Prep`, which encodes specific prepositions, is incorporated along with all the other features. This grammar is much larger than the first version, due to the large number of prepositions that can be values of the feature `Prep`. Both VP and NP rules get multiplied by the number of prepositions included.

4.5.2 Training Data for inside-outside estimation

The training data consists of unannotated Wall Street Journal data. The training data set is first tagged using Treetagger (Schmid, 1994)[3]. We only select sentences in the Wall Street Journal corpus that are ≤ 25 words, in order to keep estimation time within limits of the available computational resources. Average sentence length in this corpus is 15.47 words. Experiments are carried out with training corpora of 4, 8 and 12 million words. Each training corpus C is divided into n subparts $C_1 ... C_n$, based on the number of processors available for a given experiment.

[3] A few tag names used by Treetagger need to be translated into the corresponding tag names used in our treebank PCFG models.

Table 4.2: List of lexical features on PCFG categories

Category	Feature	Values
Verbs (VB, VBD, VBN, VBG, VBP, VBZ)	Valence Vsel Sbj Prep	aux,b,bn,bnp,bp,br,d,de,dn,dnr,dr,e,en, ep,m,mp,n,np,npr,nr,ns,nt,p,pr,ps,r,rs,rt,s,t,z fin,base,n,h,g,to,sc,scclr, -, wnn,wnns,wnnp,wjj,wpos e, ei,t, - about, after, against, among, although, as, at, because, before, between, by, for, from, if, in, into, like, of, on, over, since, than, that, through, to, under, until, whether, while, with,-, _
Nouns (NN, NNS, NNP, NNPS)	Valence Prep	pp, s, sbar lgs, with, on, of, in, from, for, by, as, to, -,_
Adverbs (RB)	adverb attachment	s, vp, np
Adjectives (JJ)	EM class feature	4 classes

4.5.3 Test Data

We use three test-sets for evaluation of the re-estimated models. Since the initial model is a treebank PCFG, the re-estimated models contain context-free productions of the same form as the treebank PCFG. Hence parses os held-out sentences obtained using re-estimated models can be compared to treebank gold standard parses. One of the mo-

tivating factors for using treebank PCFGs for all our experiments with inside-outside was indeed that the treebank trees provide an excellent gold standard for evaluation of unsupervised models, which is also comparable with results reported by other researchers. Two such testsets (Testset I and II) are described below. The third testset that we use is the standard test section (section 23) of the Penn Treebank.

Testset I

We are interested in evaluating how the re-estimated models represent the subcategorization frames for verb that are *unseen* in the treebank. We construct a testset (Testset I) for the purpose of this evaluation as follows: First, we select 117 verbs whose frequency in Penn Treebank sections 0-22 is between 10-20[4] . We consider these verbs to be "mid-frequency" verbs, typical examples of open-class words, neither too common nor too rare. All sentences containing occurrences of these verbs are held out from the training data before building the treebank PCFG. The effect of holding out these sentences is to make these 117 verbs *novel* (i.e. unseen in training). These sentences form Testset I. The testset contains 1210 sentences, with 1258 token occurrences of the test verbs. It is used to evaluate the learning of subcategorization frames of novel verbs. Although no criteria other than frequency range were applied in selecting the test words, they have fairly varied subcategorization patterns. Table 4.3 shows all subcategorization frames associated with test verbs in this testset, along with their frequency, obtained from the

[4]The words are: *accomplish acquires admit aims aired arguing asserts auctioned betting boasts buoyed combining completing concede converting cooperate coordinate coupled decides declaring defeated defended demanded demanding denying deserve disagree discontinued eating evaluate examine exceeding exceeds execute explore export fails fare favors firmed gathering grows guarantee having hearing hinted hiring hitting honor imposing inched indicted influenced integrated interpreted justify labeled lending leveraged locked loved lowering marketed matching meaning mention miss missed monitored name observed obtaining occurs opens owes pegged plead proposes pulling rally receives recommend recommending relied resign reviewing ride rushed satisfy scuttle searching signaled slash slipping spoke spur spurred stick supports switched taped tendered tends threw track transform treated trim troubled understands upset voting waive wear wins withdrawn yielding.*

Table 4.3: The frequencies of subcategorization frames of all test verbs in Testset I (gold standard), illustrating the variety in frames in the testset. The gold standard frames are extracted from feature structure trees from sentences in Testset I

Subcat. Frame	Frequency in Testset I	Relative Freq (%)
n.-.-	662	52.62
b.-.-	124	9.86
np.-.-	121	9.62
z.-.-	115	9.14
p.-.-	73	5.8
s.e.to	50	3.97
nr.-.-	21	1.67
s.e.g	12	0.95
dn.-.-	11	0.87
d.-.-	10	0.79
m.-.-	9	0.72
r.-.-	7	0.56
de.-.-	6	0.48
s.e.sc	6	0.48
s.-.to	5	0.4
t.-.-	5	0.4
s.-.-	5	0.4
s.-.sc	3	0.24
e.-.-	3	0.24
pr.-.-	2	0.16
bn.-.-	2	0.16
aux.-.g	1	0.08
npr.-.-	1	0.08
s.-.fin	1	0.08
s.-.g	1	0.08
s.-.n	1	0.08
bp.-.-	1	0.08

gold standard feature-trees of sentences in the testset – it is seen that the testset contains verbs with a variety of frames with a typical Zipfian distribution.

Table 4.4: The occurrence frequency in gold standard trees of Testset II, of subcategorization frames of all verbs.

Frame	Frequency	Rel. Freq	Frame	Frequency	Rel. Freq.
n.-.-	3780	32.29	npr.-.-	12	0.1
t.-.-	1134	9.69	aux.-.-	7	0.06
z.-.-	999	8.53	aux.-.wjj	7	0.06
b.-.-	990	8.46	br.-.-	7	0.06
aux.-.n	700	5.98	s.t.sc	6	0.05
np.-.-	604	5.16	s.-.n	5	0.04
aux.-.h	518	4.43	en.-.-	5	0.04
p.-.-	516	4.41	aux.-.wnn	5	0.04
s.e.to	495	4.23	rs.e.to	5	0.04
aux.-.g	376	3.21	ns.-.fin	4	0.03
s.-.-	197	1.68	aux.-.to	3	0.03
aux.-.base	145	1.24	rs.e.g	3	0.03
nr.-.-	138	1.18	bp.-.-	3	0.03
d.-.-	136	1.16	ep.-.-	3	0.03
de.-.-	109	0.93	s.e.wnn	2	0.02
dn.-.-	105	0.9	mp.-.-	2	0.02
s.-.to	102	0.87	ns.-.-	2	0.02
s.-.sc	74	0.63	ps.-.-	1	0.01
m.-.-	68	0.58	s.t.n	1	0.01
e.-.-	53	0.45	rt.-.-	1	0.01
s.e.sc	52	0.44	bnp.-.-	1	0.01
s.e.g	48	0.41	t.-.fin	1	0.01
r.-.-	48	0.41	ns.e.g	1	0.01
bn.-.-	40	0.34	s.e.wjj	1	0.01
s.-.base	37	0.32	np.e.to	1	0.01
s.-.fin	31	0.26	s.-.wnn	1	0.01
ns.e.to	25	0.21	rs.-.-	1	0.01
aux.-.fin	21	0.18	ps.e.to	1	0.01
s.e.base	20	0.17	ns.t.to	1	0.01
pr.-.-	19	0.16	ns.e.base	1	0.01
t.e.to	18	0.15			
s.-.g	14	0.12			

Testset II

We also construct another testset (Testset II) by holding out every 10^{th} sentence in Penn Treebank sections 0-22 to give a total of 4310 sentences. This test set is not specially constructed in any way, and is larger than section 23. It is used to evaluate overall accuracy of subcategorization error. We cannot use section 23 of the Penn Treebank for evaluation of subcategorization frame acquisition for the following reason: section 23 was held out from the context-free grammar used as the backbone of the feature grammar. Thus the feature-constraint grammar may not contain the rules required to convert treebank trees into feature-structure trees. Since features in the feature-structure tree for a test sentence is used to create the gold standard for the subcategorization frames of verbs in the test sentence, section 23 cannot be used as a test set to evaluation subcategorization acquisition. Testset II sentences however were not held out from the feature-constraint grammar, but are held out while constructing the PCFG. We therefore have available feature-structure trees with the relevant features for sentences in this testset. These can be used as gold standard for evaluation of subcategorization frames of verbs in test Viterbi parses.

Table 4.4 shows the distribution of subcategorization frames of all verbs in Testset II. Since Testset II includes all verbs, auxiliaries and light verbs while Testset I (Table 4.3) contains only mid-frequency open-class verbs, this distribution is different from the one in Table 4.3.

4.5.4 Parameters for the Lexical Transformation

The transformation interleaved between iterations of inside-outside was described in Chapter 3, Equation 3.7. The value of λ in the lexical transformation is 0.5 for all

the following experiments, except when otherwise noted, giving equal weight to the treebank and the re-estimated models.

4.6 Re-estimation with a 4 million word training corpus: standard and interleaved procedures

4.6.1 Interleaved re-estimation

A series of six models were obtained by carrying out six iterations of the interleaved inside-outside re-estimation procedure, with a smoothed treebank model *t* as the starting model and with 4 million words of training data. The PCFG version is numbered **b14**. The re-estimation was carried out at the Cornell Theory Center (now the Cornell Center for Advanced Computing), in parallel on about 50 dual-processor, 3.6 GHz Xeon machines with 4 GB RAM. Each iteration with 4 million words of data took approximately 15-20 hours. The re-estimation was done on a Windows platform using Bitpar (Schmid, 2004), compiled to run under Cygwin (a linux-like environment for Windows).

4.6.2 Standard inside-outside re-estimation

In order to evaluate models obtained using the interleaved procedure against comparable models obtained using standard inside-outside, a separate series of four models were obtained by carrying out four iterations of standard inside-outside re-estimation, using the same treebank PCFG as a starting model and the same training data as in the interleaved procedure.

4.6.3 Labeled Bracketing results

As a basic evaluation of the quality of the re-estimated models, we report the labeled bracketing scores on the standard test section (section 23) of the Penn Treebank (see Table 4.5). Using the re-estimated models, maximum probability (Viterbi) parses are obtained for all sentences in section 23, after stripping away all treebank annotation, including the pre-terminal tag. The labeled recall, precision and f-scores for parses with re-estimated grammars from successive iterations are shown under columns It 1, It 2, etc. in Table 4.5. The baseline model t_{0t} is created by smoothing the treebank model t_0 with the test data from section 23[5]. Test data from section 23 must be merged in with the treebank model in order to accommodate unknown words from section 23. This is done as described in Chapter 2, §2.9 in Equation 2.3, with the test words from section 23 (tagged using Treetagger) forming $g(w, \tau)$ and $\lambda = 0.1$. A testset is always merged with a given model in this manner before parsing, to account for unknown words. This baseline is relevant if one wants to measure the improvement of re-estimated models over the treebank-trained model. Our other baseline is the smoothed treebank model with the unlabeled training corpus merged in, t. This is the model used as the initial model for re-estimation and thus is the real baseline for the re-estimation procedure. In practice, the two baselines are very close. We had expected that t would be a little worse than t_{0t}, since some frequency is taken away from the treebank model t_0 and spread out amongst the large training corpus. In practice, the f-score for this model is marginally better. This model is in effect smoothed twice – once with a value of $\lambda = 0.001$ while merging the unannotated training corpus, and again with $\lambda = 0.1$ while merging the test corpus before parsing.

[5]The labelled bracketing scores of this model are slightly lower than that reported in Chapter 2 due to holding out an additional ~ 6000 sentences from the treebank training set.

Table 4.5: Labeled Bracketing scores for various models, on Penn Treebank Section 23. p-values are calculated between model t_{0t} and re-estimated models.

	t_{0t}	t	It 1	It 2	It 3	It 4	It 5	It 6
Interleaved Procedure								
Recall	86.48	86.49	*86.72	*86.79	*86.79	*86.78	86.81	86.72
Precision	86.61	86.63	*86.95	*87.07	*87.06	*87.07	87.04	87.01
f-score	86.55	86.56	*86.83	***86.93**	*86.92	*86.92	86.92	86.86
p-values								
Recall			<0.022	<0.005	< 0.008	<0.008		
Precision			<0.003	<0.00001	<0.0003	<0.0002		
Standard Procedure								
Recall	86.48	86.49	87.95	87.11	86.42	85.55	-	-
Precision	86.61	86.63	85.99	84.79	83.37	82.06	-	-
f-score	86.55	86.56	**86.96**	85.93	84.87	83.77	-	-
p-values								
Recall			<0.025	=0.119				
Precision			=0.128	<0.0005				

Labeled bracketing scores: Interleaved Procedure

All models obtained using the interleaved procedure (up to 6 iterations) show an improvement over both baselines. The best model is the one obtained after 2 iterations, after which the score is reduced a little. All models from the interleaved procedure surpass state-of-the-art *unlexicalized* PCFG scores on Wall Street Journal data; the highest *f*-score for unlexicalized PCFGs on Section 23 is 86.6% in Schmid (2006), to the best of our knowledge. The improvement is significant ($p < 0.005$ for recall and $p < 0.0001$ for precision, for the best model).

Labeled bracketing scores: Standard Inside-outside

Table 4.5 also shows scores for grammars estimated using the standard inside-outside procedure. The first re-estimated model is better than any model obtained from either procedures. However, there is a large disparity in precision and recall scores. Recall is much higher than the treebank baseline and is approaching the Collins (1997) recall value of 88.6 for a lexicalized model. Precision, however, is below the baseline level. In the second iteration, the *f*-score falls below the baselines, dropping by more than a percent point. *f*-scores deteriorate successively in the next few iterations, with precision faring worse than recall. This result is not surprising – inside-outside estimation is known to result in suboptimal PCFG models. The comparative f-scores of the standard inside-outside re-estimation and the interleaved procedure justifies the constraints imposed on the models in the interleaved procedures, v.i.z., using syntactic parameters from the treebank and lexical transformations that merged re-estimated lexical models with the treebank model. Even with a good initial model such as the smoothed treebank model, standard inside-outside produces models that quickly get worse after the first iteration in the absence of any other constraints[6].

Statistical Significance Test

The significant test used to determine whether differences in parsing scores are statistically significance is known as "stratified shuffling", a form of randomized test. In this test, the null hypothesis is that the two models that produced the observed results are the same, such that for each test instance (in this case, a sentence that was parsed), the two

[6]In the standard inside-outside procedure reported here with WSJ training data, syntactic parameters in the re-estimated model that had zero-frequency were removed from the model. In another version of the experiment on New York Times data, we maintained these parameters in the model, by giving them a very small frequency. The results were largely the same, with parsing f-scores of models quickly deteriorating after the first iteration.

observed scores are equally likely. This null hypothesis is tested by randomly shuffling the scores of individual sentences between the two models and then re-computing the evaluation metrics (precision and recall, in this case). After a large number of iterations n, the likelihood of incorrectly rejecting the null hypothesis is $(nc+1)/(nt+1)$, where *nc* is the number of random differences greater than the original observed difference, and *nt* is the total number of iterations. The program used is written by Dan Bikel and is available from the author's website *http://www.cis.upenn.edu/ dbikel/*.

4.6.4 Evaluation of subcategorization learning

We measure the error rate in the identification of the subcategorization frame of 1360 tokens of 117 verbs in Viterbi parses of sentences from Testset I obtained using re-estimated models. Recall from §4.5.3 that these verbs are novel verbs with respect to the treebank model. Table 4.6 shows this error rate (i.e. the fraction of test items which receive incorrect tag-incorporations in Viterbi parses) for various models obtained using the interleaved and standard re-estimation procedures. The error rate is obtained as described in §4.3.1.

t_{0t1} is the model with the test data from Testset I merged in (to account for unknown words) using the smoothing scheme given in Equation 2.3. This model has no verb specific information for the test verbs. Each test verb has a smoothed subcategorization frame (SF) distribution proportional to the SF distribution for all verbs in the lexicon with that tag. This baseline error is about 33.4%. This means that there is enough information in the average distribution of all verbs and in the surrounding syntactic context, to assign the correct subcategorization frame to a novel verb in a Viterbi parse in about 66.6% of cases. For the models obtained using the interleaved re-estimation,

Table 4.6: Subcategorization error for novel verbs (Testset I).

Iteration *i*	Interleaved Procedure	Standard Procedure
t_0	33.36	33.36
1	*24.40	28.69
2	*23.45	25.56
3	*23.05	27.86
4	*22.89	28.41
5	**__22.81__	-
6	*22.83	-

the error rate falls to the lowest value of 22.81 % for the model obtained in the 5th iteration: an absolute reduction of 10.55 points, and a percentage error-reduction of 31.6%. All improvements of the interleaved procedure over the baseline are statistically significant (>99.9% confidence, using the McNemar test for matched pairs). Much of the improvement in error rate occurs with the first iteration. However, the error rate for the model obtained in the 5^{th} iteration is significantly better than the model obtained in the 1^{st} iteration (> 95% confidence)(shown with two stars), showing that the models continue to improve from the first to the fifth iteration. The error rate increases slightly in the sixth iteration. It is possible that the models improve even after the sixth iteration after a temporary rise in the error rate – we stopped re-estimation at this point due to computing resource constraints, and also since improvements, if any, after the sixth iteration would be quite small in any case.

Standard inside-outside

The models obtained using standard re-estimation do not perform as well as the ones from the interleaved procedure on the task of subcategorization detection in Viterbi parses. Errors for novel verbs are shown in Table 4.6. Even for the model from the first iteration, whose labeled bracketing score on section 23 was the highest out of all re-estimated models (refer to Table 4.5), the SF error is higher than the corresponding model from the interleaved procedure (possibly due to the low precision of this model). The error rate for the standard procedure starts to increase after the second iteration (in contrast to the interleaved procedure) mirroring the decrease in f-score of these models.

4.6.5 Re-estimated Lexicons

We examine the lexical entries of some test verbs in the re-estimated lexicons. Table 4.7 shows the development of lexical entries for three representative test verbs in four iterations of the interleaved procedure. These verbs are novel verbs (i.e., they do not occur in the un-smoothed treebank lexicon). The frequencies shown in Table 4.7 are scaled according to the formulas in §4.5.4 (absolute frequencies in the unsupervised training sample are higher) and only the top five most-frequent subcategorization frames for each verb are shown.

Table 4.7: Lexical entries (top 5 SFS) for three novel test verbs in successive iterations. The frequencies are scaled. The last column shows the distribution of these verbs in a treebank model where they were not held-out. The verb tags are VB (base), VBP (non-3rd-person present tense) and VBZ (3rd person present tense). Interpretation of valences (those in the column for the fourth iteration) are b (that-clause), n (transitive), p (prepositional), s.e.to (control) and z (intransitive).

t	It 1	It 2	It 3	It 4	PTB
disagree					
VBP.n.-.- 0.0014	VBP.z.-.- 2.01	VBP.z.-.- 2.20	VBP.z.-.- 2.22	VBP.z.-.- 2.23	VBP.z.-.- 1.0
VBP.t.-.- 0.0012	VBP.p.-.- 0.98	VBP.p.-.- 1.17	VBP.p.-.- 1.20	VBP.p.-.- 1.22	VBP.p.-.- 1.0
VB.n.-.- 0.0011	VB.p.-.- 0.64	VB.z.-.- 0.60	VB.z.-.- 0.61	VB.z.-.- 0.61	VB.z.-.- 1.0
VBP.aux.-.h 0.0009	VB.z.-.- 0.56	VB.p.-.- 0.54	VB.p.-.- 0.53	VB.p.-.- 0.53	VB.b.-.- 1.0
VBP.b.-.- 0.0008	VBP.n.-.- 0.27	VBP.n.-.- 0.27	VBP.n.-.- 0.25	VBP.n.-.- 0.24	-
admit					
VB.n.-.- 0.0040	VB.n.-.- 2.06	VB.n.-.- 2.12	VB.n.-.- 2.13	VB.n.-.- 2.16	VB.n.-.- 0.5
VB.z.-.- 0.0009	VBP.b.-.- 1.27	VBP.b.-.- 1.49	VBP.b.-.- 1.48	VBP.b.-.- 1.48	VBP.p.-.- 0.5
VBP.n.-.- 0.0008	VB.b.-.- 0.63	VB.b.-.- 0.99	VB.p.-.- 0.32	VB.b.-.- 0.76	VB.z.-.- 0.5
VBP.t.-.- 0.0007	VBP.n.-.- 0.44	VB.p.-.- 0.32	VBP.n.-.- 0.31	VB.z.-.- 0.33	VBP.z.-.- 0.5
VB.t.-.- 0.0006	VB.z.-.- 0.33	VBP.n.-.- 0.32	VB.z.-.- 0.30	VB.p.-.- 0.32	-
decides					
VBZ.t.-.- 0.0014	VBZ.b.-.- 1.28	VBZ.s.e.to 1.14	VBZ.s.e.to 1.16	VBZ.s.e.to 1.16	VBZ.s.e.to 3.5
VBZ.n.-.- 0.0011	VBZ.s.e.to 0.90	VBZ.b.-.- 1.09	VBZ.b.-.- 1.04	VBZ.b.-.- 1.06	VBZ.b.-.- 1.5
VBZ.aux.-.h 0.0008	VBZ.z.-.- 0.63	VBZ.n.-.- 0.37	VBZ.n.-.- 0.36	VBZ.n.-.- 0.36	VBZ.n.-.- 0.5
VBZ.b.-.- 0.0006	VBZ.s.-.fin 0.42	VBZ.z.-.- 0.32	VBZ.z.-.- 0.35	VBZ.z.-.- 0.35	VBZ.p.-.- 0.5
VBZ.z.-.- 0.0005	VBZ.n.-.- 0.36	VBZ.p.-.- 0.29	VBZ.p.-.- 0.28	VBZ.p.-.- 0.28	-

The first column is the smoothed treebank model, which contains an average distribution over all frames for these verbs as a result of smoothing (there is no verb-specific distribution for these verbs in the treebank model since they have been removed from the treebank training data). The frequency distributions in the initial model are different for each verb since the average treebank distribution is multiplied by a term corresponding to the scaled corpus frequency for the verb (see Equation 4.1). The columns labeled It1 to It4 are frequency distributions from re-estimated models (interleaved procedure) from iterations 1 to 4. The last column shows the gold distribution, obtained by constructing a treebank model which included treebank sentences containing these verbs (scaled by 0.5 to be comparable to the rest of the columns, since λ in Equation 3.7 (lexical transformation) is 0.5). The movement of the frames for each verbs can be observed across the iterations. The distribution in the column labeled It4 is considered to be the final "learnt" verb-specific distribution.

The treebank frequency of these verbs is too small to make a quantitative evaluation meaningful. However, by examining the top five frames for each verb in the column It4, it does seem that the frames are reasonable frames for the verb in question – all or most of the frames for each verb appear in the treebank (seen under the column labeled PTB). For example, for the verb *decides*, the It4 column contains frames `s.e.to`, `b`, `n` and `p`, all of which appear in the treebank (under column PTB). Contrast this with the frames for *decides* in the first column *t*, which do not resemble the frames for the verb in the treebank. Notice that the frame `aux.-.h` present under *t*, which is certainly wrong since *decides* is not an auxiliary verb, has been eliminated in the It4 distribution. The same is true of the other two verbs as well– the frames seen in column It4 have a much closer match to the frames under column PTB, as compared to frames in initial model *t*, which do not match up against the ones seen in the treebank for these verbs.

Table 4.8: 4M words training data, 4300 test sentences (Testset II). Statistically significant reductions are marked with a * (> 99.9% confidence) and ** (> 95% confidence).

TB Freq.	type/token	t_0	It 1	Abs.	% Reduc.
all	2793/13467	18.5	16.84	1.66	*8.97
0	411 / 412	41.26	33.03	8.25	*20.00
1	208 / 208	32.69	24.52	8.17	*24.99
2	145 / 145	36.55	22.76	13.79	*37.73
3	150 / 173	26.59	19.08	7.51	*28.24
4	131 / 144	22.22	20.83	1.39	6.26
5	118 / 134	24.63	19.40	5.23	*21.23
6	97 / 118	24.58	18.64	5.94	*24.17
6-10	366 / 490	22.24	19.59	2.65	**11.92
11-20	406 / 766	21.54	18.02	3.52	*16.34
21-50	428 / 1319	19.41	19.11	0.3	1.55
51-100	173 / 1147	19.44	19.09	0.35	1.80
101-200	105 / 1352	18.71	18.57	0.14	0.75
201-500	46 / 1340	23.06	22.31	0.75	3.25
501-1K	5 /321	18.07	16.82	1.25	6.92
1K-2K	5 / 816	12.38	12.25	0.13	1.05
2K-5K	5 / 1719	9.42	7.62	1.8	*19.11
$> 5K$	2 / 1224	10.54	10.13	0.41	3.89

4.7 Analysis of subcategorization learning: effect of treebank occurrence frequency

An interesting question regarding re-estimation of a treebank PCFG on unannotated data (and indeed, regarding semi-supervised learning in general) is the question of the

relative utility of annotated treebank data versus unannotated data. One expects that parameters related to words occurring with high frequency in the treebank are accurately estimated from the treebank data. For these cases, unsupervised estimation may not provide much improvement, and may even corrupt the treebank estimate. Unsupervised training is expected to be useful for cases where treebank data is not sufficient for a reliable estimate, i.e. in the case of parameters related to low frequency or novel words. In order to explore the role of treebank occurrence frequency, we divide the set of test verbs from Testset II into subsets based on their frequency of occurrence in the treebank training data. We the measure the subcategorization error for tokens of verbs belonging to each subset separately in the test data. Table 4.8 shows error rates for verbs divided into these sets. The error reduction is always calculated between the models from Iteration 1 and the baseline model since most of the error reduction occurs in the first iteration, although in some cases models from higher iterations have a smaller error.

4.7.1 Overall subcategorization error

The first row in Table 4.8 shows the SF error for all verbs in Testset II for models obtained with the interleaved procedure. The overall error reduction is 8.97% in Iteration 1(models from later iterations might have a slightly lower error). This error rate is comparable to the accuracy of other parsing and token-based evaluations reported in the literature. For example, Briscoe & Carroll (1997) report a token-based evaluation for seven verb types – their system gets an average recall accuracy of 80.9% for these seven types. The verbs they select for evaluation appear to be high-frequency verbs, although this is not clear from their paper. Their accuracy is slightly lower than the accuracy of our re-estimated model, which has an overall accuracy of 83.16% as seen in Table 4.8. For low frequency verbs (exemplars <10) they report that their results are around chance.

While there has been substantial previous work on the task of SF acquisition from corpora (Brent (1991); Manning (1993); Briscoe & Carroll (1997); Korhonen (2002), amongst others), we find that relatively few parsing-based evaluations are reported. Since their goal is to build probabilistic SF dictionaries, these systems are evaluated either against existing dictionaries, or on distributional similarity measures. Most are evaluated on *high*-frequency verbs (unlike the present work), in order to gauge the effectiveness of the acquisition strategy.

The second row of Table 4.8 is the error for verbs which have zero frequency in the treebank training data (i.e. unseen verbs): Note that this error reduction is much less than the 31.6% in Testset I. The verbs in Testset II are truly rare in the treebank and hence have much fewer token occurrences in the unlabeled corpus than the test verbs in Testset I, which were artificially made novel but are really mid-frequency verbs. Note however that although these verbs are "rare" in the Penn Treebank they really are common verbs known to most competent speakers of English[7]. Since these verbs have only a few occurrences in the unannotated training data (see Table 4.10), it is possible that the error rate will decrease further if the size of the unlabeled corpus is increased and the number of occurrences of these verbs in the training data increases correspondingly. In the case

[7]Examples are: admit, allocating, anchor, appeased, append, arbitrates, arguing, assigns, attacks, awoke, backfires, backpedaling, bandied, bangs, bargain, barreling, bartered, berated, betting, blast, blend, boating, bog, bottled, brag, brazen, buckling, buoyed, burbles, burglarized, buzzes, cadge, campaigned, canning, carp, catapult, ceases, centering, chafe, chastised, chilled, chopping, chortled, classed, combining, commemorate, commemorated, compound, compromised, computes, concede, condemns, confides, confiding, confiscating, congratulate, consenting, console, consoles, converged, converting, coordinate, correspond, coughing, counseled, counterprogram, crawls, crisscrossing, cropped, crowed, cycling, dabbled, dancing, decelerating, decides, decreases, defeated, defeating, defect, defended, defining, deformed, defuse, demanded, demanding, denying, deserve, dining, disagree, discontinue, discontinued, disembark, disseminate, dissuade, don, donned, dotting, duplicating, dwindling, eavesdrop, effects, eked, empathize, encapsulate, endeavor, ennumerated, enriching, enrolled, envy, equate, escorts, evaluate, examine, excoriated, execute, exhausting, export, expounding, extrapolated, fare, farm, favors, ferreting, ferry, filched, filtered, firmed, fizzes, flamed, flinch, flinging, floats, floundered, flouting, forbade, forged, formulated, forsaken, fragments, franchises, frequents, furloughed, galvanizing, gestured, glaze, gliding, gloats, government-set, governmentset, graduates, grimaced, grows, gunslinging, hampers, handicap, hands, hastened, hearing, hitting, hoard, honored, howling, humbled, immigrated, impelled, imposing, impounded, improvised, inched, indicted, inferred, inflating, intercepting, interpreted, interviewing, intimidate, irks, jelled, jerked, justify ...

of novel verbs in Testset II, there is almost no improvement after the 1st iteration, while for novel verbs in Testset I, the error rate continues to improve until Iteration 5. This difference between the drop in error rates for the two testsets across later iterations could be related to the fewer occurrences of Testset II verbs in the training data as compared to Testset I verbs. Chapter 3 contained an example of how EM can learn to resolve ambiguous analyses through unambiguous ones. A large number of token occurrences in the training data means that there is some variability in analyses – there are some ambiguous and some unambiguous contexts for the verb in question. It is possible that as the iterations proceed, this variability causes some analyses to differ between iterations. If there are very few occurrences of a verb in the training data, there is less variability in the analyses, and hence much less movement across the iterations.

4.7.2 Low frequency verbs

Low frequency verbs are considered to be those with an occurrence frequency up to 20 in the treebank training data (an arbitrary cut-off point). There is significant error reduction for low frequency verbs (Table 4.8). The large reduction is not hard to understand: the treebank does not provide enough data to have good parameter estimates for these verbs and the re-estimation procedure benefits parameters of these words the most.

4.7.3 Middle frequency verbs

For verbs that have a fair number of occurrences in the treebank (> 20), the benefit of the unsupervised procedure is reduced. This is seen for verbs of frequency of 20 upwards (Table 4.8). The re-estimation procedure does result in an error reduction, but it is not

Table 4.9: Very high frequency verbs in Testset II

501-1K	did, do, make, rose, say
1K-2K	been had, 's, says, were
2K-5K	are, be, has, have, was
> 5K	said, is

statistically significant. The treebank contains enough occurrences of these verbs to have a good distribution of subcategorization frames for these verbs in the treebank lexicon and re-estimation is not very useful. It is possible that for verbs in this frequency range, the smoothing procedure needs to be designed differently. These verbs have enough occurrences in the treebank that the lexical entry for the verb in the treebank model has a fairly accurate word-specific subcategorization distribution. Thus, it might be beneficial that the smoothed distribution for a verb in this category be derived from the treebank distribution of that verb alone, and not from all verbs in the treebank, as is currently done.

4.7.4 High frequency verbs

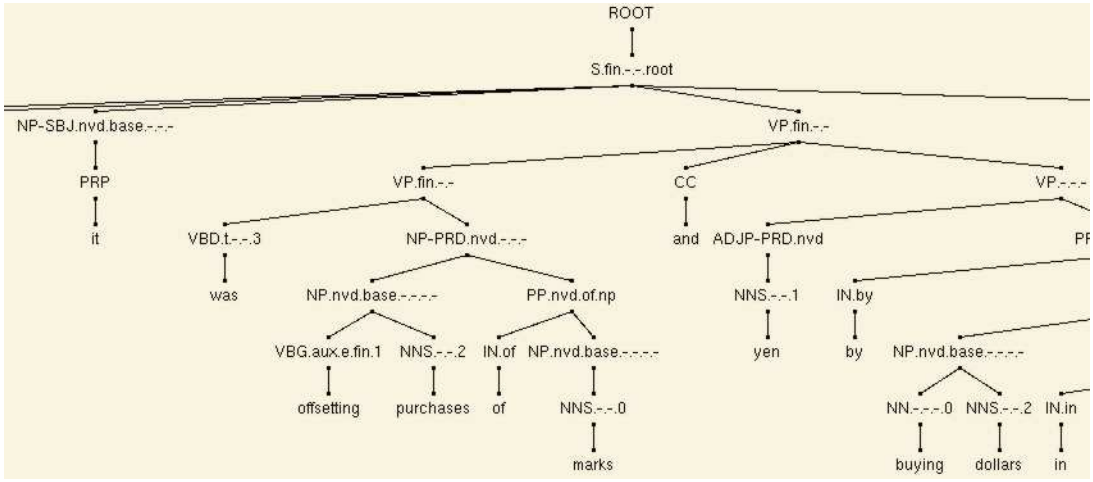
Surprisingly, we found that error reduction for some very high frequency verbs (for instance, those with more than 500 occurrences in the treebank) is also fairly high; we expected that parameters for high frequency words would benefit the least from unsupervised estimation, given that they are already common enough in the treebank to be accurately estimated from it. Out of all verbs that occur with a frequency greater than 500 in the treebank, the error reduction for the range 2000 – 5000 is statistically significant. The error reduction for the others is higher than that for mid-frequency verbs. The high frequency verbs consist of very few types – mainly auxiliaries, some light

verbs and a few others (see Table 4.9 for a list of verb types in the treebank that fall into these ranges). Since light verbs typically have a large number of frames, it is possible that the treebank estimate for these is quite sparse in spite of their high occurrence frequency, and as a result re-estimation from large data is beneficial. The frequency range 2000 – 5000 consists solely of auxiliary verbs. Examination of Viterbi parses shows that improved results are largely due to better detection of predicative frames in re-estimated models. There are several cases in the test data of a predicative frame being correctly recognized in parses with the model from Iteration1 but not the baseline model. Figure 4.7 shows better detection of an auxiliary VP. Figure 4.7 (a) shows the auxiliary verb *was* in the sentence *It was offsetting purchases of marks and yen*, being wrongly assigned a predicative frame in the baseline model (with the gerund getting an NP-PRD tag). In a parse with a re-estimated model (Figure 4.7 (b)), the auxiliary gets identified correctly, with a VP complement. The gold standard parse for the sentence showing the correct attachment is shown in Figure 4.8 for comparison. Figure 4.9 shows parses illustrating better detection of a predicative frame for an auxiliary verb. In the parse with the baseline model, the verb *are* gets an intransitive frame `z.-.-`. In the parse with the re-estimated model, it gets the correct predicative frame `t.-.-.`, with a PP-PRD complement[8], although the attachment of the locative PP (PP-LOC) is still incorrect. The golf standard parse is shown in Figure 4.10

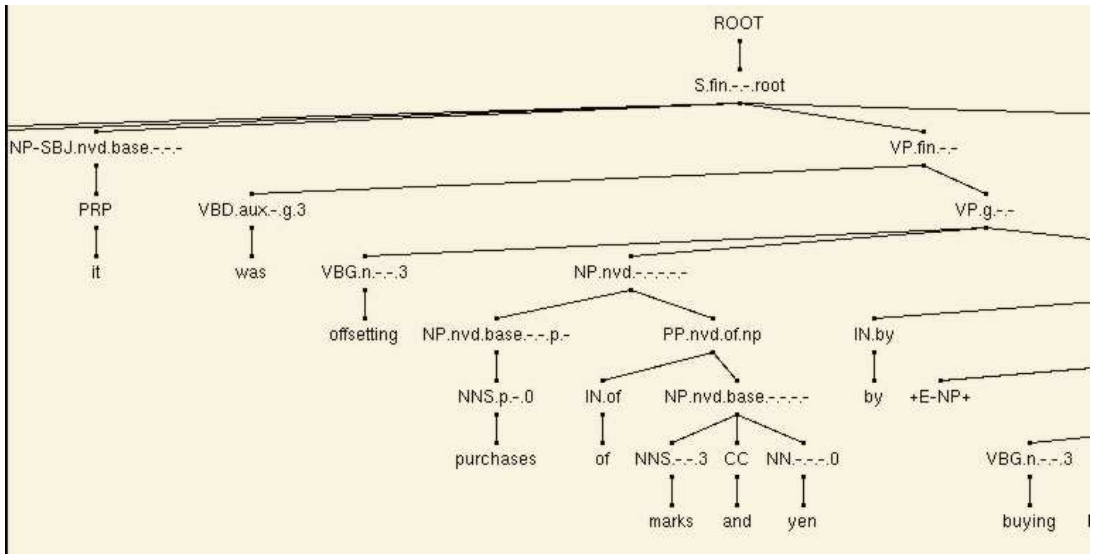
4.8 Effect of occurrence frequency in unlabeled training data

The previous section analyzed the effect of the occurrence frequency of a verb in the annotated (treebank) data on the utility of the re-estimation. It is also to be expected

[8]The PP-LOC-PRD category is not considered to be a predicate in this grammar version, something that should be rectified. Thus, this example is also an illustration of unsupervised re-estimation fixing an error that is the result of a bug in the treebank grammar.



(a) Baseline model



(b) Re-estimated model (Iteration 1)

Figure 4.7: Parses showing better detection of an auxiliary VP in the re-estimated model (b), as compared to the baseline model (a). Here, the auxiliary verb *was* is incorrectly assigned a predicative frames in the baseline model (a) with a portion of the progressive VP *offsetting purchases of marks and yen* incorrectly identified as a predicative NP (NP-PRD). In the parse with the re-estimated model (b), the verbs is correctly identified as an auxiliary verb, with its complement being a progressive VP.

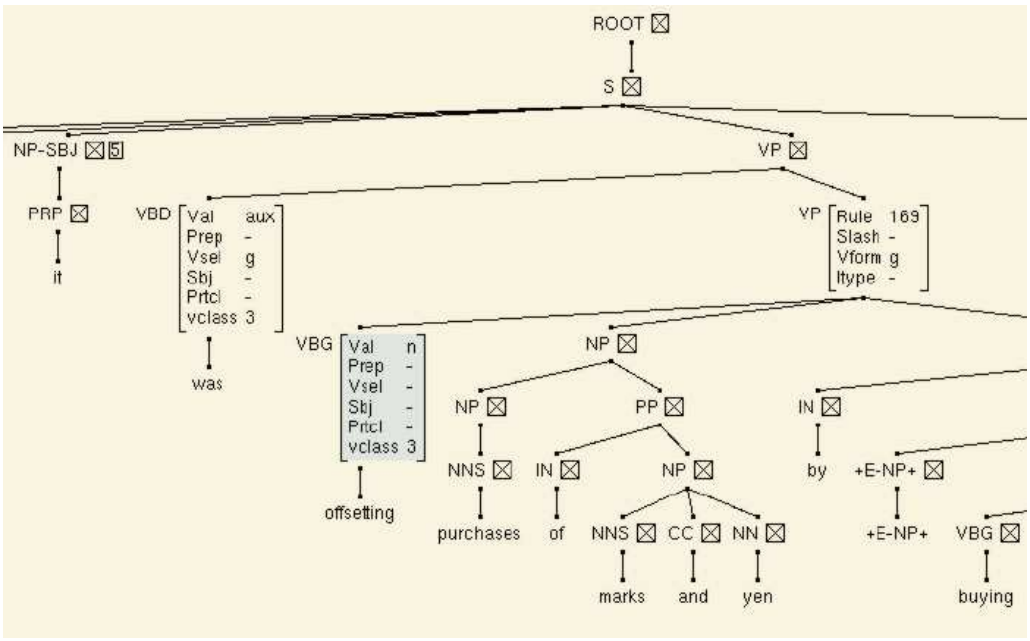
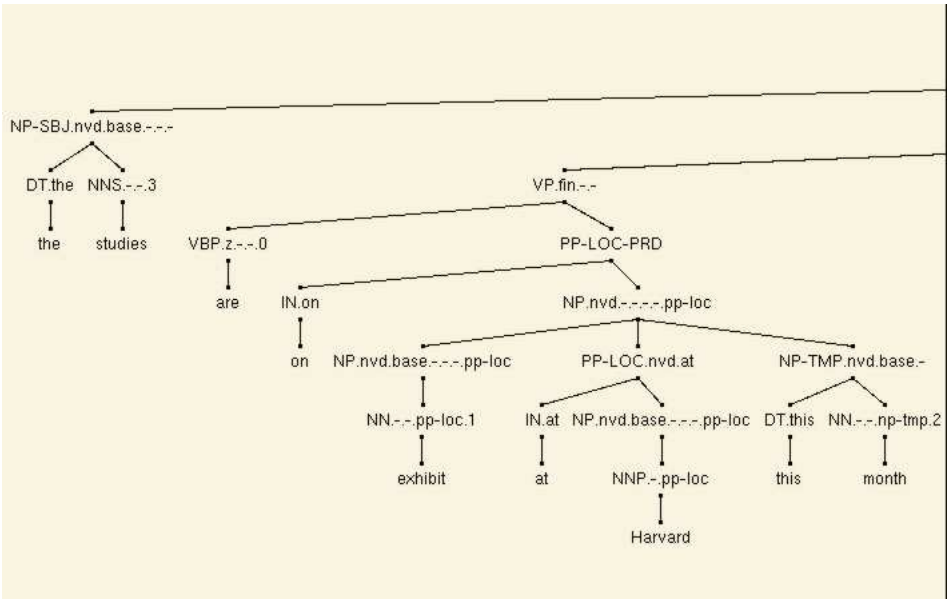


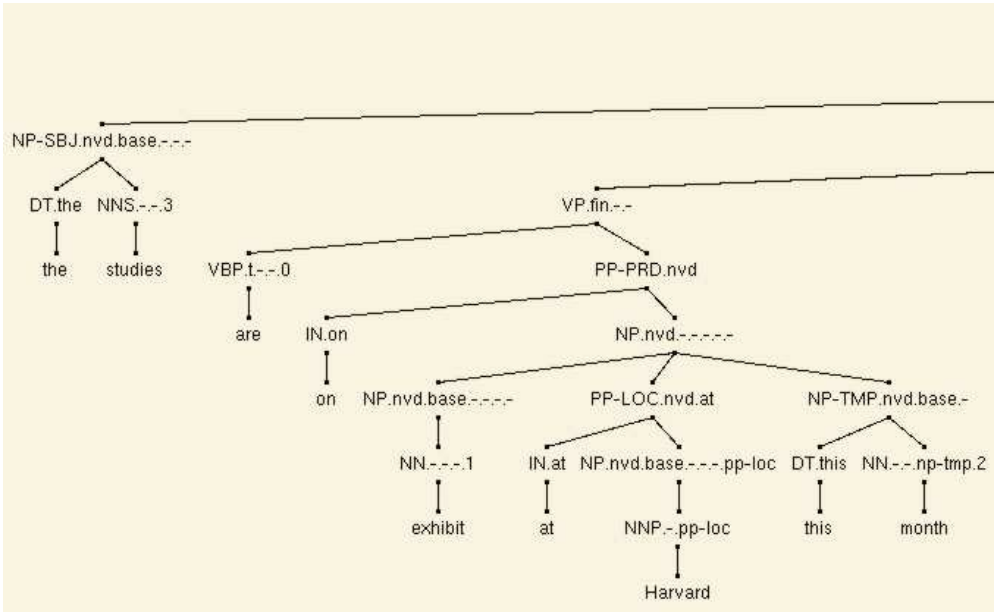
Figure 4.8: Gold standard feature-structure tree for the parses in Figure 4.7, illustrating correct identification of a progressive VP

that the accuracy of the learnt SF distribution for a verb is dependent on the number of occurrences of the verb in the unannotated training data. We therefore analyze subcategorization error by breaking up the set of test verbs by their occurrence frequency in the unlabeled training data.

We expect that the average occurrence frequency f_{train} of a word in unlabeled WSJ training data is the same multiple of its treebank occurrence frequency f_{tb} as the size of the training data is to the size of the treebank. In our case, the training data of 4 million words is 4 times the size of the treebank. We first verify that the ratio holds true in our training sample – in Table 4.10, we see that, for low frequency verbs, the average occurrence frequency in the training data of a set of verbs of a given treebank occurrence frequency is approximately 4 times their treebank frequency. Table 4.10 also shows (in the last column) the *range* of occurrence frequency in training data f_{train} for words of treebank frequency 1 to 10. This frequency range is very large, which means that for a



(a) Baseline model



(b) Re-estimated model (Iteration 1)

Figure 4.9: Parses showing better detection of a predicative frame. In (a), the baseline mode, the verb *are* is incorrectly assigned an intransitive frame `z.-.-.`, while in the re-estimated model (b), it is correctly assigned a predicative frame `t.-.-`.

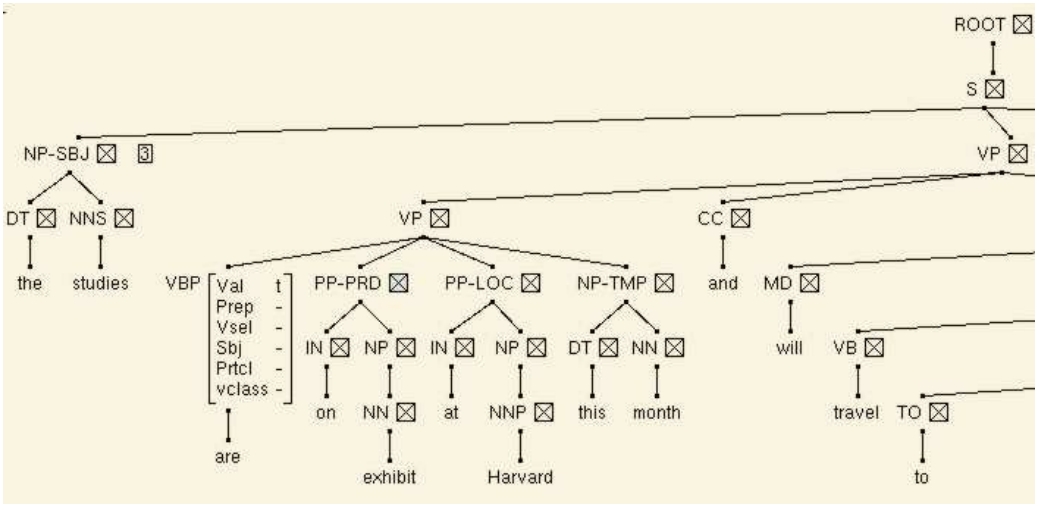


Figure 4.10: The gold standard feature-structure tree for the sentence in Figure 4.9, illustrating correct attachment of a progressive VP.

Table 4.10: The relation between occurrence frequency in treebank and unlabeled data

TB Verb Freq. $freq_{tb}$	Average Freq. in Training Data (4 Million words) $freq_{train}$	Range in Training Data
1	4.07	0-93
2	7.98	0-52
3	12.39	0-47
4	17.67	0-255
5	20.75	0-84
6-10	31.7	0-90

given subset of test verbs with a particular frequency of occurrence in the treebank, there are some verbs which occur with a much higher ratio than 4 in the unlabeled training data while there are others which have a zero or very low occurrence count. In order to examine if the training data occurrence frequency is correlated to the reduction in subcategorization error, it is useful to break up the set of test verbs further, according to their occurrence frequency in the training sample. In Table 4.11, we take test verbs (from Testset II) belonging to a particular treebank frequency range – columns in Table 4.11 represent the treebank frequency. The same subsets as in Table 4.8 are used; we examine only low frequency verb sets since this problem only applies to low-frequency words. The rows in Table 4.11 represent occurrence frequency in the unlabeled training sample; in each row, we eliminate some verbs which have occurred in the training data above or below a certain frequency. The first row in Table 4.11 shows the error rate for all verbs in the subset of a given treebank frequency. This error rate is the same as that shown in Table 4.8 for each subset. The rest of the rows show different frequency ranges in the unannotated training data, for treebank frequencies from 1-20. Thus, the row labeled > 0 represents subcategorization error for test verbs which have occurred at least once in the training sample, > 1 represents error for verbs with at least two occurrences in the training sample, and so on. We do not show verbs of treebank frequency 4 since the re-estimated models do not have a significant improvement for these verbs (see Table 4.8). Comparing rows of different f_{train} frequencies to the row for all verbs, we see that the error rate drops as we eliminate test verbs with a very low number of occurrences in the training sample. For example, verbs which have more than 20 occurrences ($f_{train} > 20$) have a much lower error rate than the rate for the entire set of verbs.

For verb-sets of treebank frequency 5 and more, it appears that the pattern changes a bit: a large number of training occurrences is not beneficial. Notice that for treebank frequency 5, the error rate for verbs which occur with frequency < 10 is much lower

Table 4.11: Breakup of low frequency treebank verbs, according to their occurrence frequency in training data: Subcategorization error for these subsets.

f_{train}	TB Freq. 1	TB Freq. 2	TB Freq 3	TB Freq 5	TB Freq 6	TB Freq 6-10	TB-Freq 11-20
all	23.56	24.14	20.23	19.4	18.6	19.88	17.35
> 0	22.84	24.31	20.35	23.88	18.6	19.88	17.35
> 1	23.33	22.30	19.64	23.88	18.6	19.88	17.35
> 2	22.29	21.64	19.51	24.24	18.6	19.88	17.35
> 5	22.12	19.30	19.50	24.62	18.8	20.04	17.35
> 10	18.75	22.37	18.10	25.41	18.5	20.3	17.41
> 20	8.33	17.86	18.52	20.99	20.9	21.67	17.23
> 30	-	-	4.55	19.51	25.5	22.18	16.91
> 40	-	-	-	-	-	-	-
> 50	-	-	-	-	18.71	18.71	
< 50	-	-	-	-	-	18.38	14.62
< 20	-	-	-	24.4	-	13.59	23.81
< 10	-	-	-	12.5	-	-	-

than the overall error rate for this set. The pattern looks similar for verbs of treebank frequency 6, 6-10 and 11-20. It is not obvious why this might be the case: one explanation could be that for verbs that have an abnormally large number of occurrences in the training data, there could be a particular valence frame that is predominant in the particular corpus which skews the distribution. It is not possible to run statistical tests on these error rates, since dividing the test verbs into subsets, first by treebank frequency and then by training data frequency leads to each subset having very few types and tokens.

4.8.1 Breakup of subcategorization error by Frame Type

The previous section analyzed the effect of annotated and unannotated training frequencies on learning accurate subcategorization frame distributions (as evaluated by verbs being assigned correct frames in Viterbi parses). It is also interesting to analyze subcategorization frame learning based on the type of frames being learnt. Our grammar has some fairly complex frame types such as `s.e.sc`, `s.e.to`, `s.-.to`, etc, in addition to basic types like `n.-.-`, `z.-.-` etc. We would like to know, for instance, if there are some frames that are easier to acquire than others. In this section we explore this aspect. We perform analyses over Testset I, which contains open class and unseen verbs only, and also separately on Testset II, which has a more typical distribution of seen and unseen verb types that includes auxiliaries and light verbs.

Table 4.12 shows the subcategorization error for each frame type for novel verbs in Testset I, for the baseline model *t*, and for models obtained from the first iteration of the interleaved procedure. It also shows the error reduction for each frame type. Ignoring frames which have less than 5 tokens in the testset, where the statistics are unreliable, it is seen that several different frame types have a large amount of reduction in error. The error reduction is not limited to a few high frequency frames, but is seen for a variety of frames types across a range of occurrence frequencies. One must note that for the common frame types like `n.-.-.-` there are presumably a much larger number of occurrences in the training data. Thus, we expect that there is a larger opportunity for learning the more common frames. However, frames of a lower frequency also seem to have a fair error reduction, indicating that frames being learnt are not limited to only a few out of the possible frames in the lexicon.

An interesting point is that there is an improvement in frame types `np.-.-.-` and `p.-.-.-`. There is a general opinion in the field of statistical treebank parsing that

Table 4.12: Subcategorization error by frame type for model obtained in Iteration 1 of the interleaved procedure, for novel verbs in Testset I. Verbal tags are considered to be part of our subcategorization frame, but are removed here.

Frame	# tokens (Gold)	%Error t	%Error It1	%Error Reduction
n.-.-	662	23.87	18.73	21.52
z.-.-	115	38.26	33.91	11.36
np.-.-	121	34.71	32.23	7.14
p.-.-	73	27.4	20.55	25
b.-.-	124	12.1	12.1	0
s.e.g	12	83.33	58.33	30
d.-.-	10	90	80	11.11
nr.-.-	21	38.1	33.33	12.5
s.e.to	50	16	12	25
m.-.-	9	77.78	33.33	57.14
dn.-.-	11	63.64	54.55	14.29
de.-.-	6	66.67	66.67	0
r.-.-	7	42.86	71.43	-66.67
s.-.to	5	40	40	0
s.e.sc	6	33.33	0	100
e.-.-	3	66.67	33.33	50
pr.-.-	2	50	50	0
npr.-.-	1	100	100	0
bp.-.-	1	100	100	0
s.-.n	1	100	100	0
s.-.fin	1	100	100	0
s.-.sc	3	33.33	33.33	0
bn.-.-	2	0	50	-

PP-CLR annotations in the Penn Treebank are inconsistent and hence unreliable. Our subcategorization annotation in the treebank considers all PPs with the -CLR functional tag to be complements. The improvement in these frame types indicates that PP-CLR marking in the Penn Treebank is consistent, at least to some extent.

Testset II

Next, we examine the error reduction for verbs in Testset II – most verbs in this testset have been seen in the treebank and also include auxiliaries and light verbs. Table 4.13 shows the recall error rates for subcategorization identification in this testset, while Table 4.14 shows precision errors. Only frames with a token occurrence of more than 5 in the testset are reported. We notice that the pattern is not very different from that of novel verbs in Testset I. A large number of frame types have a large and positive error reduction, including some frames for auxiliary verbs like `aux.-.n` and `aux.-.g`.

There is not a substantial disparity between precision and recall error rates for either the baseline model or the re-estimated model.

4.8.2 Analysis of errors in subcategorization identification for novel verbs

We also perform an analysis to understand the type of errors that occur in the identification of the most common subcategorization frames. Table 4.15 shows errors in the detection of the transitive frame (only the top ten error categories are shown). Table 4.16 shows the top ten errors in the identification of the `z` (intransitive), `np` (NP-PP) and `p` (prepositional) frames. These frames are the three most frequently-wrong frames, after the transitive frame (see Table 4.12).

Transitive frames are most commonly mis-analyzed as NP-PP frames (Table 4.15). Similarly, in Table 4.16, NP-PP frames are most commonly mis-analyzed as transitive frames. This is not surprising; these errors arise either from PP attachment mistakes, a well-known problem for PCFGs, or a mistaken analysis of a PP as an argument/non-

Table 4.13: Subcategorization Frames with their recall error for re-estimated models, on all verbs from Testset II (only frames with 5 or more occurrences in the testset are reported)

Frame	Gold Tokens	% err (t)	% err (It 1)	% reduction
n.-.-	3780	16.32	15.05	7.78
t.-.-	1134	11.82	11.73	0.75
z.-.-	999	35.24	31.53	10.51
b.-.-	990	7.98	7.27	8.86
aux.-.n	700	4.43	2.71	38.71
np.-.-	604	38.58	34.27	11.16
aux.-.h	518	4.63	1.74	62.5
p.-.-	516	28.29	28.68	-1.37
s.e.to	495	9.09	6.87	24.44
aux.-.g	376	4.26	3.72	12.5
s.-.-	197	15.74	14.21	9.68
aux.-.base	145	4.14	3.45	16.67
nr.-.-	138	31.16	24.64	20.93
d.-.-	136	39.71	37.5	5.56
de.-.-	109	13.76	13.76	0
dn.-.-	105	61.9	57.14	7.69
s.-.to	102	12.75	10.78	15.38
s.-.sc	74	24.32	21.62	11.11
m.-.-	68	32.35	29.41	9.09
e.-.-	53	26.42	22.64	14.29
s.e.sc	52	25	21.15	15.38
s.e.g	48	20.83	18.75	10
r.-.-	48	47.92	50	-4.35
bn.-.-	40	35	32.5	7.14
s.-.base	37	18.92	18.92	0
s.-.fin	31	48.39	48.39	0
ns.e.to	25	36	40	-11.11
aux.-.fin	21	90.48	90.48	0
s.e.base	20	15	15	0
pr.-.-	19	57.89	42.11	27.27
t.e.to	18	72.22	66.67	7.69
s.-.g	14	28.57	35.71	-25
npr.-.-	12	75	75	0
br.-.-	7	28.57	28.57	0
aux.-.-	7	85.71	85.71	0
aux.-.wjj	7	100	100	0
s.t.sc	6	16.67	16.67	0

Table 4.14: Precision error in SF frames for re-estimated model (Iteration 1), on all verbs from Testset II (only frames with 5 or more occurrences in the testset are reported)

Frame	Gold Tokens	% err (t)	% err (It 1)	% reduction
n.-.-	3780	15	14.13	5.82
t.-.-	1134	11.11	11.11	0
z.-.-	999	32.83	30.23	7.93
b.-.-	990	7.78	7.17	7.79
aux.-.n	700	4.43	2.71	38.71
np.-.-	604	38.41	34.27	10.78
aux.-.h	518	4.63	1.74	62.5
p.-.-	516	27.13	27.71	-2.14
s.e.to	495	8.69	6.67	23.26
aux.-.g	376	4.26	3.72	12.5
s.-.-	197	15.74	14.21	9.68
aux.-.base	145	4.14	3.45	16.67
nr.-.-	138	29.71	23.91	19.51
d.-.-	136	38.24	36.76	3.85
de.-.-	109	13.76	13.76	0
dn.-.-	105	60.95	57.14	6.25
s.-.to	102	12.75	10.78	15.38
s.-.sc	74	24.32	21.62	11.11
m.-.-	68	32.35	29.41	9.09
e.-.-	53	26.42	22.64	14.29
s.e.sc	52	25	21.15	15.38
r.-.-	48	47.92	50	-4.35
s.e.g	48	18.75	18.75	0
bn.-.-	40	32.5	30	7.69
s.-.base	37	18.92	18.92	0
s.-.fin	31	45.16	45.16	0
ns.e.to	25	36	40	-11.11
aux.-.fin	21	90.48	90.48	0
s.e.base	20	15	15	0
pr.-.-.-	0	0	0	
t.e.to	18	66.67	61.11	8.33
s.-.g	14	28.57	35.71	-25
npr.-.-	12	75	75	0
aux.-.-	7	85.71	85.71	0
aux.-.wjj	7	100	100	0
br.-.-	7	28.57	28.57	0
s.t.sc	6	16.67	16.67	0

Table 4.15: Top ten misanalyses of transitive frames: Novel verbs in Testset I

Gold Frame	Test Frame	err (*t*)	err (It 1)	err (It 2)
n.-.-.-				
	np.-.-	50	45	33
	t.-.-	35	3	6
	aux.e.fin	12	11	8
	z.-.-	10	6	7
	s.-.sc	7	5	5
	s.-.to	6	5	1
	n.-.-	4	7	7
	ns.e.to	4	4	6
	dn.-.-	4	4	5
	m.-.-	3	2	1

argument of the verb. Thus a PP which is attached correctly but not identified as an argument (i.e., as a PP-CLR) will be counted as an error.

Intransitive frames are also most commonly mis-analyzed as PP frames, and vice-versa, demonstrating the inability to distinguish between an argument and an adjunct PP, i.e., adjunct PPs attached to the verb are being marked as arguments, with the verb being assigned a PP frame (`p.-.-`).

4.9 Interleaved re-estimation with larger training data

In the previous experiments, unlabeled training data of size 4 million words was used for re-estimation of the treebank PCFG. Although inside-outside estimation is a com-

Table 4.16: Top ten misanalyses of `z.-.-.-` (intransitive), `np.-.-.-` (NP-PP) and `p.-.-.-` (prepositional) frames: Novel verbs in Testset I

	Test Frames	EM0	EM1	EM2
z.-.-:				
	p.-.-	12	16	16
	t.-.-	10	5	0
	b.-.-	6	4	5
	n.-.-	6	5	5
	s.e.to	3	2	2
	aux.e.fin	2	3	2
	r.-.-	1	1	1
	s.-.-	1	0	0
	nr.-.-	1	1	0
	s.e.fin	1	0	0
np.-.-:				
	n.-.-	22	21	24
	aux.e.fin	6	4	3
	t.-.-	4	3	3
	s.-.sc	3	0	1
	dn.-.-	2	6	7
	de.-.-	2	0	0
	s.e.sc	1	0	0
	m.-.-	1	0	0
	np.e.to	1	0	0
	b.-.-	0	1	1
p.-.-:				
	z.-.-	6	3	5
	t.-.-	4	2	2
	d.-.-	3	0	0
	p.-.-	2	2	2
	b.-.-	1	0	0
	ps.e.to	1	1	1
	r.-.-	1	0	0
	bp.-.-	1	0	0
	nr.-.-	1	2	
	d.-.-	0	4	5

putationally expensive process, placing a practical restriction on the amount of data that can be used for training, 4 million words of training data is a fairly small data set for unsupervised methods. In comparison to the treebank data, it is only about 4 times in size. In order to examine the effect of more unannotated data, we also ran the interleaved re-estimation procedure using unannotated corpora of $\sim$ 8 and $\sim$ 12 million words.

Table 4.17 shows the subcategorization error for novel verbs in Testset I for models obtained using the interleaved procedure, for training data of different sizes. The error in the second iteration for the grammar estimated with 8 million words of training data is lower than the corresponding error of models estimated with 4 million words. The error rate of 22.26% for 8 million words in Iteration 2 is statistically significantly lower than the best error rate of 22.81% with 4 million words (in the fifth iteration). Training data of 12 million words resulted in a further reduction of error rate to 21.86% in the third iteration. The difference in error between models obtained in Iteration 2 with 8 million words and Iteration 3 with 12 million words is not statistically significant; however, the error rate with 12 Million training words is still dropping. We were not able to run further iterations with this amount of training data due to limitations in computing resources.

Table 4.19 compares the subcategorization error of models obtained with 8 million words of training data with the error rate for models obtained with 4 million words (for subsets of test verbs in Testset II divided according to treebank occurrence frequency). For most of the low frequency verbs, training with 8 million words results in improved error rates over the model obtained with 4 million words. * indicates a significant difference over the baseline of t_{0t}. Although models with 8 million words gave better error rates than models with 4 million words, the differences between the two were not statistically significant.

Table 4.17: Subcategorization error for novel verbs (Testset I): grammars re-estimated using 4 M, 8 M and 12 M words.

Iteration *i*	Subcat Error 4M words	Subcat Error 8 M words	Subcat Error 12 M words
t	33.47	33.47	33.47
1	*24.40	*24.64	*24.46
2	*23.45	****22.26**	*22.80
3	*23.05	*22.34	***21.86**
4	*22.89	*23.05	
5	****22.81**	-	
6	*22.83	-	

Table 4.18: Labeled Bracketing f-score for re-estimated grammars using 4, 8 and 12 M words of training data.

Iteration *i*	f-score 4M words	f-score 8 M words	f-score 12 M words
$t_0 t$	86.55	86.55	86.55
t_t	86.56	86.56	86.56
1	86.83	86.86	86.80
2	*86.93	86.82	*86.84
3	*86.92	*86.89	86.80
4	*86.92	*86.89	-
5	86.92	-	-
6	86.86	-	-

Table 4.19: Subcategorization error breakup by frequency range: Testset II (4300 sentences) and 8 M words of training data.

TB Freq.	t_0	It 1 8M	Abs. Reduc.(8M)	% Reduc 8M	%Reduc 4M
1	32.69	23.08	9.61	29.4	24.99
2	36.55	22.76	13.79	37.72	37.73
3	26.59	19.08	7.51	28.24	28.24
4	22.22	20.83	1.39	6.25	6.26
5	24.63	19.40	5.23	21.23	21.23
6-10	22.24	19.18	3.06	13.75	11.92
11-20	21.54	17.62	3.92	18.19	16.34
21-50	19.41	18.65	0.76	3.91	1.55
51-10	19.44	19.44	0	0	1.80
101-200	18.71	18.34	0.37	1.97	0.75
201-500	23.06	22.16	0.90	3.9	3.25
501-1K	18.07	17.13	0.37	5.20	6.92
1K-2K	12.38	11.52	0.86	9.64	1.05
2K-5K	9.42	7.33	1.58	22.18	19.11
>5K	10.54	10.38	0.16	1.51	3.89

Labeled bracketing f-scores for models obtained with larger training data (8 and 12 million words) does not match the f-scores obtained by models estimated on 4 million words, although there is significant improvement over the baseline treebank grammar for each. Table 4.18 shows comparative f-scores for models obtained with the three training data sizes, on Section 23. The f-score for the model obtained in the second iteration with 12 million words is statistically significantly better than the baseline of t_0. The fact that f-scores of models estimated on larger training data do not show a corresponding

increase may be a reflection of the fact that some accurate treebank estimates (such as those related to high-frequency words) are getting corrupted due to the re-estimation. Thus, overall parse quality is worsened, although structures selected by novel verbs show an improvement.

4.10 Subcategorizing for specific prepositions

One of the long-standing issues in parsing has been the attachment of prepositional phrases to verbal or nominal nodes. The prepositional head of a PP plays an important role in attachment of the PP to the verb or noun, i.e., verbs and nouns select for particular prepositions (Hindle & Rooth, 1993). We saw in the previous section that NP and PP frames are being identified better in parses by re-estimated models. Re-estimation from large data might have even more benefit for PP attachment if specific prepositions are a part of the subcategorization frame of verbs. We therefore built a new version of the treebank PCFG (Version b15) in which specific prepositions are incorporated into verbal categories for subcategorization frames that include a Prepositional Phrase (like `np`, `p`, etc.). Thus, the full verbal subcategorization frame for this version of grammar contains four features `Val`, `Vsel`, `Sbj` and `Prep` (see Table 4.2 for the list of all included features and prepositions). The grammar also incorporated specific prepositions into the nominal subcategorization frame for NN and NNS categories (common nouns). The nominal subcategorization frames contain a feature marking nominal valence (with four values: `p` for prepositional, `s` for S complements, `sbar` for SBAR complements, and a default value).

Re-estimation was carried out with the interleaved procedure and with 4 million words of training corpus. Labeled bracketing scores are shown in Table 4.20. Incorpo-

Table 4.20: Labeled Bracketing f-score (Section 23) for re-estimated grammar incorporating specific prepositions in verbal and nominal valence (Grammar version b15).

Iteration *i*	f-score b14	f-score b15
t	86.56	86.48
1	86.83	86.88
2	*86.93	*86.89
3	*86.92	-
4	*86.92	-
5	86.92	-
6	86.86	-

rating prepositions into verbal and nominal categories has the disadvantage of splitting each VP and NP rule into several rules (each rule is split into rules equal to the number of prepositions incorporated, in our case about 30 prepositions). This splitting makes frequencies estimated from the treebank too sparse, reducing the performance of the treebank-trained grammar. This is seen in the slight lowering of the baseline f-score for the smoothed treebank grammar *t* in Table 4.20. The re-estimated models show an improvement in f-score over the baseline, with the improvement in the model obtained from Iteration 2 being statistically significant (p-value for recall difference: 0.00519, precision difference: 9.99e-05). It is possible that the models for Version b15 continue to improve; we were not able to run more iterations at this time. Inside-outside estimation with version b15 takes much more computing resources as compared to estimation with version b14, due to the much larger number of rules (both syntactic and lexical) in b15.

Table 4.21 compares the subcategorization errors for re-estimated models for ver-

Table 4.21: Subcategorization error for novel verbs: b14 and b15 grammars with 4M words of EM training data.

Iteration *i*	Subcat Error **b14**	Subcat Error **b15** (excl. Prep)	Subcat Error **b15** (incl Prep.)
t	33.47	34.18	34.98
1	24.40	24.96	25.52
2	23.45	24.40	25.04
3	23.05	-	
4	22.89	-	
5	22.81	-	
6	22.83	-	

sions b14[9] and b15, for novel verbs in Testset I. The two columns for version b15 correspond to a subcategorization frame that excludes specific prepositions (middle column) and includes them (last column). Note that models of Version b15 always have the preposition incorporated into verbal and nominal categories during re-estimation and parsing; the middle column in Table 4.21 represents the error rate when the preposition is ignored during *evaluation* of the Viterbi parse. The baseline error for version b15 is higher than that for version b14 in both cases, i.e. when the preposition is part of the evaluated frame and when is ignored in the evaluation. In the case when the preposition is included in the evaluation, the subcategorization frame is more detailed and hence more difficult to identify correctly. In the case when the preposition is not included in the frame, the grammar model is still sparser than b14, and hence performs worse. However, there is a substantial reduction in subcategorization error in re-estimated models in both cases.

[9]Version b14 contains no prepositions in verbal and nominal subcategorization frames.

We also examine the effect of incorporating specific prepositions on the cross-errors between transitive frames (NP complements) and NP-PP complements. An analysis of Viterbi parses of sentences in Testset II shows that the error for NP-PP frames `np-.-.-` is lower for a treebank model that does incorporate specific prepositions in the subcategorization frame of verbs and nouns. This is seen in the first row in Table 4.22. This error corresponds to the total number of tokens in the testset which have an NP-PP frame in the gold standard, but erroneously get a different frame in the Viterbi parse. The second row in Table 4.22 shows the number of errors for re-estimated models of the two grammar versions; the number of errors in NP-PP frames decreases in the case of the version which does not incorporate specific prepositions (version b14), while the error increases with re-estimation for the grammar which incorporates specific prepositions (version b15).

Table 4.23 shows the number of NP-PP frames that are wrongly analyzed as transitive frames – this error is due to argument PPs either not being attached correctly as sisters of the verb or not being recognized as an argument (as a PP-CLR) if they are attached correctly. In this case too, the pattern in the same as before. The treebank-trained model with prepositions incorporated into verbal categories does better than the treebank-trained model without prepositions. However, re-estimated models with prepositions do worse than re-estimated ones without prepositions.

It is not apparent why re-estimation of a model that incorporates prepositions results in np-pp cross errors getting worse. More analysis is require to understand this result. One possibility is that the sparseness of the initial grammar and lexicon makes them unable to impose good constraints on the re-estimated model. Another issue is that prepositions were introduced in both verbal and nominal frames at the same time. It will be interesting to conduct re-estimation with prepositions incorporated only in verbal

Table 4.22: Subcategorization frame errors in Testset II Viterbi parses for np subcategorization frame (NP - PP).

	# errors	# errors
	Version b14 No Prep. in SF	Version b15 Prep in SF
Baseline model (smoothed treebank model *t*)	233	103
Re-estimated model (It. 1)	207	121

Table 4.23: Number of np (NP-PP) subcategorization frames erroneously detected as n (transitive) frames, in Testset II Viterbi parses.

	# errors	# errors
	Version b14 No Prep. in SF	Version b15 Prep in SF
Baseline model (smoothed treebank model)	177	81
Re-estimated model (It. 1)	149	95

frames. The NP rules in this case will not be as sparse as in version b15, possibly affecting the outcome of re-estimation.

4.10.1 Learning noun valence

Statistical distributions for valences of nouns is an area which, like verbal subcategorization, is afflicted by sparseness in treebank data, and thus would benefit from re-estimation over a larger corpus. Our grammar contains a valence feature nval incorpo-

Table 4.24: Noun Valence errors for grammar b159 with 4M words of training data.

Iteration *i*	Noun valence Error Version b15
0	23.13
1	20.35
2	21.49

rated into nouns (NN and NNS categories), along with a Preposition feature `nvalperp` for incorporating specific prepositions (only in version b15). We therefore perform a preliminary analysis of noun valence on re-estimated models of version b15. As usual, noun valence features on Viterbi parses were compared to those on corresponding feature-structure trees for the held-out Testset II. The total number of noun tokens (of type NN and NNS) in the testset is 9138, out of which a large number has the default valence "-" in the gold trees. Discounting those tokens with the default value, the number of tokens with `p`, `s` or `sbar` valence is 1405. Table 4.24 shows the reduction in identifying these for the baseline and re-estimation models. There is a small reduction in error for the re-estimated model. Clearly, learning nominal valence is an area that warrants more experimentation, but is beyond the scope of this dissertation.

4.10.2 Effect of the interpolation parameter λ in the lexical transformation

The formula used for lexical transformation in the above experiments was described in Chapter 3, and is repeated below for convenience. The value of λ is set to 0.5 for all POS tags τ and sequences of incorporated features ι, giving equal weight to the treebank

and the re-estimated models.

$$d_i(w, \tau, \iota) = (1 - \lambda_{\tau,\iota})t(w, \tau, \iota) + \lambda_{\tau,\iota}\bar{c_i}(w, \tau, \iota) \tag{4.3}$$

The analysis of subcategorization errors of middle to high frequency verbs suggested that it might be beneficial to give more weight to the treebank as compared to the re-estimated model while interpolating parameters for middle and high frequency words. These words already have a high enough occurrence frequency in the treebank model that parameters associated with them will be fairly well estimated. Re-estimating these parameters from unannotated data might cause more harm than benefit. Thus, we carried out some experiments where the value of the interpolation parameter λ is different for words in different frequency ranges, as shown below.

Let $t(w)$ be the treebank frequency of a word w,

$$\begin{aligned}
&\text{if } t(w) <= 5 , && \text{let } \lambda_f = x \text{ , } x = 0.5 \\
&\text{if } 5 < t(w) <= 15 , && \lambda_f = x/2 \\
&\text{if } 15 < t(w) <= 50 , && \lambda_f = x/10 \\
&\text{if } t(w) > 50 , && \lambda_f = x/100
\end{aligned} \tag{4.4}$$

The values in Equation 4.4 were empirically chosen, with limited experimentation, by smoothing treebank PCFG models using these values, and measuring the f-score of Viterbi parses on test data. These particular values improved f-score over the original treebank model, showing that a smoothing scheme that is sensitive to treebank frequency of words is a good idea. In order to find the optimal values of these smoothing parameters, more experimentation is required.

We ran one iteration of the interleaved procedure by using values of λ as shown in

Table 4.25: Labeled bracketing F-score for models smoothed using different values of λ for different frequency ranges (Test data: Section 23 of PTB).

	Original Models ($\lambda = 0.5$)	Models with λ_f as in Eq. 4.4
t	86.56	85.94
It 1	86.83	87.29

Eq. 4.4. We also smoothed the treebank model t_0 using λ values in 4.4 in order to obtain the initial model t for inside-outside re-estimation. This appears to make the treebank model worse, as seen by the drop in f-score for the baseline model t (column 2 in Table 4.25). The model obtained after the first iteration of inside-outside is merged with the treebank model using values of λ as in Eq. 4.4. The resulting model, when used to parse data from Section 23 of the Penn Treebank, has a labeled bracketing *f*-score better than the corresponding value for a model obtained using the original lexical transformation (where λ is independent of treebank frequency). The jump in f-score of the model re-estimated in the first iteration as compared to the baseline is also much larger than the corresponding increase for the model obtained using the original lexical transformation.

4.11 Chapter Summary

This chapter reported the results of re-estimating a treebank-trained PCFG model with additional data from the unannotated portion of the Wall Street Journal, using procedures based on the inside-outside algorithm. We focused on learning lexical parameters of the PCFG. Models estimated with the modified inside-outside procedure interleaved with a lexical transformation between iterations of inside-outside and retaining syntactic parameters from the treebank-trained PCFG performed better than models re-estimated

using the standard inside-outside algorithm. We obtained a very large (and statistically significant) error reduction on subcategorization error for novel verbs. We also obtained a large and statistically significant reduction of subcategorization error for verbs that occur with low frequency in the treebank. Using a larger corpus for training seems to result in better models to a certain extent: the subcategorization error for novel verbs significantly improves with the size of the training corpus. However, there is not a corresponding increase in f-score. We believe that this could be because some portions of the combined model (obtained from interpolation between treebank and re-estimated models) are getting worse from re-estimation over larger data, even though some parameters might have better estimates. Some experiments with using a different interpolation parameter between the treebank and the re-estimated lexicons indicate that the f-scores and the subcategorization errors in Viterbi parses are sensitive to this parameter. In particular, more weight should be given to the treebank model for mid or high frequency lexical items. It is also possible that mid-frequency lexical items should be scaled using relative frequencies of tag-incorporation combinations for the particular lexical entry, rather than the average tag-incorporation distribution over the entire lexicon. We analyzed errors in specific subcategorization frames: several frames show improvements; it is not the case that only a few easy but frequent frames are being acquired. We also perform some analyses that show that incorporating specific prepositions into the subcategorization frame improves the misanalysis between NP and NP-PP frames, indicating that it leads to better PP attachment. However, re-estimation with a more detailed subcategorization frame makes the cross-errors between NP and NP-PP frames worse. We do not have an explanation for this; more analysis and examination of re-estimated models is required. We also measure noun subcategorization error on common nouns; re-estimated models show a small improvement in noun subcategorization. This is also an area that warrants more exploration.

All the evaluations reported in this chapter are against gold standard data from the Penn treebank (either Penn Treebank trees, or feature-structure trees with the same tree shape as the corresponding Penn Treebank tree). This makes the evaluations of unsupervised models not only objective but also interpretable according to the annotation standards of the Penn Treebank and is an important aspect of our methodology.

CHAPTER 5

CONCLUSIONS AND FUTURE RESEARCH

This dissertation presented a method for building a linguistically-sophisticated but well-performing treebank PCFG which modeled linguistic phenomena like long-distance dependencies and argument-adjunct distinctions via the presence of traces and other null elements, Penn Treebank functional tags and additional linguistic features incorporated into the PCFG category symbols. We then presented a method for estimating the complex lexical parameters of this PCFG from unannotated data via the inside-outside algorithm. Our motivation for building such a PCFG was to model complex linguistic phenomena in a computationally simple formalism with well-understood estimation methods. It is clear that a sophisticated PCFG that models complex linguistic phenomena is still subject to limitations of the context-free formalism and lacks some of the theoretical considerations that motivate more complex theories like CCG or LFG. However, we are now able to use full-fledged estimation methods like maximum-likelihood estimation in order to learn better values from unannotated data for complex parameters in the PCFG.

The dissertation thus weaves together several issues related to grammar learning: the first is our belief that it is important for certain linguistic phenomenon to be modeled in statistical grammars. Second is the need to obtain better values for complex lexical parameters which are extremely scarce in annotated data. Third is the important question of constraining unsupervised estimation in a systematic manner. Lastly, but perhaps most importantly, is the issue of systematic evaluation and interpretation of models obtained by unsupervised estimation. For this reason, we worked with a linguistically-sophisticated Penn Treebank PCFG. The subcategorization frames and other features in our PCFG are based solely on annotations in the Penn Treebank and hence have a systematic interpretation according to the annotation guidelines of the Penn Treebank.

Thus the fine-grained, wide-coverage and probabilistic lexical resource we obtain via estimation over unlabeled data is also aligned with structures in the Penn treebank and may be translated to representations in other grammar formalisms with Penn Treebank probabilistic grammars.

Chapter 2 of this dissertation described the design of a feature-structure grammar, which was used to transform Penn Treebank trees to feature-structure trees. We also described a method to construct an unlexicalized treebank PCFG from the feature-structure treebank with linguistic features incorporated into PCFG category symbols. The features are fairly simple and small in number. The performance of the PCFG on Wall Street Journal parsing is comparable to the state-of-the-art in unlexicalized PCFGs, with a labeled bracketing f-score of 86.6% and empty category recovery with 83% accuracy on Section 23 of the Penn Treebank. Grammar development was stopped when the PCFG performance reached state-of-the-art, in order to use the treebank PCFG as the initial model and baseline for unsupervised re-estimation. It is conceivable that additional grammar development along similar lines will result in a better performing PCFG. There are several large non-terminal categories (for example, SQ, ADJP, S-ADVP, SINV) for which the grammar contains largely vanilla treebank rules. Adding constraints to these rules will certainly be beneficial.

The methodology of building the Penn Treebank PCFG is such that it can be extended to other languages with existing treebanks in the Penn Treebank style for the creation of PCFGs in these languages with fine-grained linguistic features. We believe that it will be an advantage for aspects of linguistic research as well as research on high-end parsing, that features such as inflectional category, lemmas, sub-classification of clausal categories, subcategorization frames of verbs and nouns, localized information about long distance dependencies, etc. can be made overtly available in treebank

annotations and incorporated into PCFG symbols.

Chapter 3 described a procedure for re-estimation of a treebank PCFG via inside-outside estimation. Re-estimated models were constrained by using parameter values from the treebank model by introducing a transformation between iterations of inside-outside.

Chapter 4 described empirical evaluations of the idea presented in Chapter 3. We carried out a standard and modified inside-outside procedure with a treebank PCFG as the initial model. We performed parsing-based evaluations of models obtained from inside-outside estimation on held-out data from the Penn Treebank. Evaluations on the detection of correct subcategorization frames in Viterbi parses showed re-estimation to be very useful for novel and low-frequency verbs. Re-estimation which was constrained by using syntactic and lexical parameters from the treebank model was found to outperform standard inside-outside re-estimation.

Subcategorization frames in our grammar are fairly complex and re-estimation shows a benefit across a variety of frames. Increasing the size of training data was advantageous for subcategorization acquisition, but not for labeled bracketing performance: subcategorization error reduces for models trained on 12 million words of training data as compared to 4 million words.

Labeled bracketing scores obtained by parsing section 23 of the Penn Treebank show a small but statistically significant improvement. The interleaved procedure gives better labeled bracketing performance than standard inside-outside. The improvements in labeled bracketing scores are encouraging, given the received wisdom that EM does not give positive results for PCFG induction. It is not surprising that improvements in labeled bracketing are small. Firstly, PARSEVAL scores are known to be insensitive to sub-

categorization detection (Carroll et al., 1998). Secondly, re-estimated models have an advantage over the treebank model only in the case of novel or low-frequency lexical parameters. Since these lexical items have very low token occurrences in the test data (Section 23), improvements in these parameter values do not have much of an impact. An evaluation that is more targeted to low-frequency items (such as the subcategorization evaluation described above) is therefore more meaningful.

5.1 Future Research

5.1.1 Smoothing

One of the areas in which further research is warranted is the lexical transformation or smoothing procedure applied to re-estimated models. Currently, the parameter λ which determines the relative weights given to the treebank and the re-estimated models is set to 0.5. This parameter must be made sensitive to the frequency of lexical items. For instance, for lexical items that occur with a high frequency in the treebank, higher weight must be given to the treebank distribution as compared to lexical items that occur sparsely. Preliminary experimentation has shown a small improvement in labeled bracketing scores by increasing the weight given to the treebank model for high treebank-frequency lexical items. In order to smooth the distribution of treebank words, we assign them all possible feature sequences with a distribution calculated over all words in the treebank. It might be beneficial to smooth high-frequency treebank words by a word-specific distribution, rather than average distributions as is the case now. In general, it appears that re-estimation is beneficial for certain parameters, while for others, the treebank model is more accurate – gaining a better understanding of this

balance is an interesting research question and will help to maximize the benefits of re-estimation.

5.1.2 Domain adaptation

The method described in this dissertation has been used successfully for re-estimating PCFGs using unannotated in-domain data. It is almost perfectly tailored to the task of parser-adaptation. In parser adaptation, typically, one has an accurate parser in the source domain which has to be adapted to a target domain with little or no annotated resources. The utility of lexical information for parser-adaptation is well-appreciated. For example, Lease & Charniak (2005) showed how existing domain-specific lexical resources from a target domain (part-of-speech tagging, named-entities, dictionary collocations) may be leveraged to improve parsing on the target domain. More recently, (Hara et al., 2007) specifically evaluate the impact of lexical knowledge from the target domain. In order to adapt an HPSG parser trained on the Penn Treebank to a biomedical domain, they re-train a probabilistic model of lexical entry assignments on words in the target domain and then incorporate it into the original parser. They find that re-training a model of lexical entry assignments is more critical for domain adaptation than retraining a structural model alone. The method for PCFG improvement described in this dissertation is similar in principle to the idea of re-training a model of HPSG lexical entry assignments and incorporating it in the original parser. Lexical entries in the PCFG lexicon have many of the same features as those used in HPSG-style lexical entries, like subcategorization and slash features. Given the success of this idea for parser-adaptation, it is interesting to apply the current methodology to this task. (Hara et al., 2007) use annotated data from the target domain to re-train a log-linear model of lexical entry assignments (with features consisting only of word n-grams). Inside-

outside estimation will allow the use of raw data, useful for domains in which there is insufficient or no annotated data available. Full scale PCFG re-estimation for obtaining lexical parameters from the target domain, while being expensive, might better capture long-range dependencies. In order to use the current methodology for adaptation to a new domain, the only requirement is that raw training data from the new domain be tagged with POS tags in Penn Treebank II style. The tagged corpus is then merged with the PCFG to obtain the initial lexicon for PCFG re-estimation. In addition, a small corpus of annotated data must be available (or constructed) for testing purposes. These requirements are far easier to meet than having an annotated corpus available for training in the target domain.

5.1.3 Unsupervised re-estimation for other lexical categories

This dissertation focused mainly on verbal subcategorization acquisition. The verbal tags in our grammar were very fine-grained with complex subcategorization features. Our grammar also contained an impoverished valence set for nominal categories (only common noun categories NN and NNS). Re-estimated models showed a small but statistically significant improvement in overall noun valence error in Viterbi parses. It would be worth-while to apply the methodology to acquiring finer nominal valences. Other lexico-syntactic properties such as attachment properties of adverbs, and adjectival subcategorization properties are also areas in which both treebank grammar development and re-estimation experiments are warranted.

APPENDIX A

FEATURES AND THEIR VALUES IN THE FEATURE-CONSTRAINT GRAMMAR

A.1 Enumerations

Enumerations of values of all features in the feature constraint grammar described in this dissertation are listed in Table A.1. Features present on each category in the grammar, and their types (Version b15) used in the re-estimation experiment are listed in Table A.2.

Table A.1: Enumeration of feature values for feature-constraint grammar.

Names		Values
enum	RBCLASS	0,1,2,3,4,5,6,7,-
enum	NCLASS	0,1,2,3,-
enum	JJCLASS	0,1,2,3,-
enum	VCLASS	0,1,2,3,-
enum	POSS	poss,-
enum	NPTYPE	base,-
enum	DTYPE	-, these, those, this, that, a, an, the
enum	PUNC	col, scol, hyph, elip, stop, ques, excl
enum	ADV	as, not, about, so,-
enum	VDOM	vd, nvd
enum	PARENT	s, vp, adjp, advp, np, qp, -, root

Continued on next page

Table A.1 – continued from previous page

Names		Values
enum	PERCENT	perc,-
enum	UP	np-tmp, pp-tmp, pp-loc, -
enum	NVAL	p, s, sbar,-
enum	DOL	dol, -
enum	VFORM	fin, base, n, h, g, to, sc, scclr, -, wnn, wnns, wnnp, wjj, wpos
enum	VAL	aux, b, bn, bnp, bp, br, d, de, dn, dnr, dr, e, en, ep, m, mp, n, np, npr, nr, ns, nt, p, pr, ps, r, rs, rt, s, t, z
enum	ADJVAL	s,b,p
enum	PRTCL	with, upon, up, together, through, over, out, on, off, in, forward, forth, down, by, behind, back, away, aside, around, apart, along, ahead, across, about, -, _
enum	SLASH	-, n, adv, adj, p, v, s, sbar, ssbar, sbarq, nv, sinvsbar, frag, sq, advpvp, sinv, ucp, sbarsbar
enum	CTYPE	whn, ambgs, whp, -, than, dSVP, whether, whad, that, nulcmp
enum	SBJ	e,ei,t, -
enum	PERP	lgs, with, on, of, in, from, for, by, as, to, -,_
enum	PREP	-, _, about, after, against, among, although, as, at, because, before, between, by, for, from, if, in, into, like, of, on, over, since, than, that, through, to, under, until, whether, while, with

A.2 List of features on non-terminal categories

Table A.2: List of categories and their features with types in the feature-constraint grammar (Version b15)

Category		Feature Type,Value
category	-COL-	PUNC punc;
category	-PER-	PUNC endpunc;
category	DT	DTYPE dtype;
category	RB	ADV rbadv; PARENT parent; PARENT advptype; RBCLASS rbclass;
category	NP	VDOM vdom; NPTYPE nptype; POSS pos; PERCENT percent ; UP up; NVAL nval; PERP nvalperp;
category	NP-SBJ	VDOM vdom;NPTYPE nptype; POSS pos; PERCENT percent; NVAL nval; PERP nvalperp;
category	NP-PRD	VDOM vdom; NPTYPE nptype; POSS pos; PERCENT percent; ADJVAL Val;
category	NP-ADV, NP-BNF, NP-CLR	VDOM vdom;NPTYPE nptype; POSS pos;
category	NP-CLR-LOC, NP-CLR-MNR, NP-CLR-TMP	VDOM vdom;NPTYPE nptype; POSS pos;
category	NP-DIR, NP-EXT, NP-HLN, NP-HLN-LOC, NP-HLN-TMP	VDOM vdom;NPTYPE nptype; POSS pos;
category	NP-LGS	VDOM vdom; NPTYPE nptype; POSS pos;
category	NP-LOC, NP-LOC-PRD, NP-LOC-PRD-TPC	VDOM vdom; NPTYPE nptype; POSS pos;
category	NP-MNR, NP-PRD-TMP, NP-PRD-TPC, NP-PRD-TTL, NP-SBJ-TTL, NP-TMP, NP-TMP-TPC, NP-TPC, NP-TPC-TTL, NP-TTL, NP-VOC	VDOM vdom; NPTYPE nptype; POSS pos;
category	NN	PERCENT percent; UP up; NVAL nval; PERP nvalperp; NCLASS nclass;
category	NNP	UP up; NVAL nval; PERP nvalperp;

Continued on next page

Table A.2 – continued from previous page

Category		Feature Type,Value
category	NNPS	UP up; NVAL nval; PERP nvalperp;
category	NNS	UP up; NVAL nval; PERP nvalperp; NCLASS nclass;
category	VP	SLASH Slash; VFORM Vform; SBJ Itype;
category	VBP, VBN, VBG, VBZ, VBD, VB	VAL Val; PREP Prep; VFORM Vsel; SBJ Sbj; PRTCL Prtcl; VCLASS vclass;
category	MD	VAL Val; PREP Prep; VFORM Vsel; SBJ Sbj; PRTCL Prtcl;[1]
category	TO	VAL Val; PREP Prep; VFORM Vsel; SBJ Sbj; PRTCL Prtcl;
category	S	SLASH Slash; VFORM Stype; SBJ Sbj; PARENT parent;
category	S-PRP	SLASH Slash; VFORM Stype;SBJ Sbj;
category	SINV	SLASH Slash; VFORM Stype; SBJ Sbj; PARENT parent;
category	SQ	SLASH Slash; VFORM Stype; SBJ Sbj; PARENT parent;
category	SQ-PRD	SLASH Slash;
category	SQ-TPC	SLASH Slash;
category	SQ-TTL	SLASH Slash;
category	PP	VDOM vdom; PERP Perp; SLASH Slash; PARENT parent;
category	PP-CLR, PP-DTV, PP-PUT	VDOM vdom; SLASH Slash; PREP Prep;
category	PP-CLR-DIR, PP-CLR-LOC, PP-CLR-LOC-TPC, PP-CLR-MNR, PP-CLR-PRP, PP-CLR-TMP, PP-CLR-TPC, PP-LOC, PP-MNR	VDOM vdom; SLASH Slash; PREP Prep;
category	PP-TMP	VDOM vdom; SLASH Slash;
category	PP-LOC-PRD	VDOM vdom; SLASH Slash;
category	PP-DIR	VDOM vdom; SLASH Slash;
category	PP-PRD,	VDOM vdom; SLASH Slash; ADJVAL Val;
category	PP-BNF	VDOM vdom;
category	PP-DIR-PRD	VDOM vdom;

Continued on next page

[1]The features on MD and TO are irrelevant. They are not incorporated in the PCFG.

Table A.2 – continued from previous page

Category		Feature Type,Value
category	PP-EXT, PP-HLN	VDOM vdom;
category	PP-HLN-LOC	VDOM vdom;
category	PP-LGS	VDOM vdom;
category	PP-LOC-MNR	VDOM vdom;
category	PP-LOC-PRD-TPC	VDOM vdom;
category	PP-LOC-TPC	VDOM vdom;
category	PP-MNR-PRD	VDOM vdom;
category	PP-NOM	VDOM vdom;
category	PP-PRD-PRP	VDOM vdom;
category	PP-PRD-TMP	VDOM vdom;
category	PP-PRD-TPC	VDOM vdom;
category	PP-PRD-TTL	VDOM vdom;
category	PP-PRP, PP-SBJ	VDOM vdom;
category	PP-TMP-TPC	VDOM vdom;
category	PP-TPC, PP-TTL	VDOM vdom;
category	S-ADV, S-ADV-CLR	SLASH Slash;VFORM Stype;SBJ Sbj;
category	S-CLF, S-CLF-TPC	SLASH Slash;VFORM Stype;SBJ Sbj;
category	S-CLR, S-CLR-MNR, S-CLR-PRP	SLASH Slash;VFORM Stype; SBJ Sbj;
category	S-HLN	SLASH Slash;VFORM Stype;SBJ Sbj;
category	S-LGS-NOM	SLASH Slash;VFORM Stype;SBJ Sbj;
category	S-LOC, S-MNR	SLASH Slash;VFORM Stype;SBJ Sbj;
category	S-NOM, S-NOM-PRD, S-NOM-SBJ	SLASH Slash;VFORM Stype;SBJ Sbj;
category	S-PRD, S-PRD-PRP, S-PRD-TPC, S-PRD-TTL	SLASH Slash;VFORM Stype;SBJ Sbj;
category	S-PRP-TPC	SLASH Slash;VFORM Stype;SBJ Sbj;
category	S-SBJ, S-SBJ-TTL	SLASH Slash;VFORM Stype;SBJ Sbj;
category	S-TMP, S-TMP-TPC	SLASH Slash;VFORM Stype;SBJ Sbj;
category	S-TPC, S-TTL	SLASH Slash;VFORM Stype;SBJ Sbj;
category	SINV-TTL, SINV-HLN	SLASH Slash;VFORM Stype; SBJ Sbj;
category	SINV-TPC	SLASH Slash;VFORM Stype;SBJ Sbj;
category	SBAR, SBAR-ADV, SBAR-PRP, SBAR-SBJ, SBAR-TMP, SBAR-PRD	SLASH Slash; CTYPE Ctype; PARENT parent;
category	SBAR-ADV-TPC	SLASH Slash; CTYPE Ctype;

Continued on next page

Table A.2 – continued from previous page

Category		Feature Type,Value
category	SBAR-CLR, SBAR-CLR-LOC, SBAR-CLR-TMP	SLASH Slash; CTYPE Ctype;
category	SBAR-DIR	SLASH Slash; CTYPE Ctype;
category	SBAR-HLN	SLASH Slash; CTYPE Ctype;
category	SBAR-LGS-NOM	SLASH Slash; CTYPE Ctype;
category	SBAR-LOC	SLASH Slash; CTYPE Ctype;
category	SBAR-LOC-PRD	SLASH Slash; CTYPE Ctype;
category	SBAR-MNR	SLASH Slash; CTYPE Ctype;
category	SBAR-NOM	SLASH Slash; CTYPE Ctype;
category	SBAR-NOM-PRD	SLASH Slash; CTYPE Ctype;
category	SBAR-NOM-SBJ	SLASH Slash; CTYPE Ctype;
category	SBAR-NOM-TPC	SLASH Slash; CTYPE Ctype;
category	SBAR-PRD-PRP	SLASH Slash; CTYPE Ctype;
category	SBAR-PRD-TMP	SLASH Slash; CTYPE Ctype;
category	SBAR-PRD-TPC	SLASH Slash; CTYPE Ctype;
category	SBAR-PUT	SLASH Slash; CTYPE Ctype;
category	SBAR-TPC	SLASH Slash; CTYPE Ctype;
category	SBAR-TTL	SLASH Slash; CTYPE Ctype;
category	SBARQ, SBARQ-HLN, SBARQ-NOM, SBARQ-PRD, SBARQ-TPC, SBARQ-TTL	SLASH Slash; CTYPE Ctype;
category	S/NP	SLASH Slash; CTYPE Ctype;
category	ADJP-PRD	VDOM vdom; SLASH Slash; ADJVAL Val;
category	ADVP, ADVP-CLR, ADVP-MNR, ADVP-TMP	VDOM vdom; PARENT parent;
category	ADJP, ADJP-CLR	VDOM vdom;
category	ADVP-LOC, ADVP-EXT, ADVP-LOC-PRD	VDOM vdom;
category	ADVP-PRD, ADVP-PRD-TPC, ADVP-PRD-TMP	VDOM vdom;
category	FRAG, FRAG-ADV, FRAG-HLN, FRAG-TPC	VDOM vdom;
category	NAC-TTL	VDOM vdom;

Continued on next page

Table A.2 – continued from previous page

Category		Feature Type,Value
category	NX	VDOM vdom;
category	RRC	VDOM vdom;
category	UCP, UCP-ADV, UCP-CLR, UCP-EXT, UCP-LOC, UCP-PRD, UCP-TMP	VDOM vdom;
category	WHPP	VDOM vdom;
category	PRN	VDOM vdom;
category	RP	PRTCL Prtcl;
category	PRT	PRTCL Prtcl;
category	QP	DOL u; PERCENT percent ;
category	WHNP	VDOM vdom;
category	IN	PREP Prep;
category	NX-TTL	VDOM vdom;
category	JJ	JJCLASS jjclass;
category	X, X-CLF, X-DIR, X-EXT, X-PUT, X-TMP	VDOM vdom;

Table A.3: Categories with no features

-COM-	ADVP-LOC-TPC	PDT
-DOL-	ADVP-MNR-TPC	POS
-HSH-	ADVP-PRP	PRP
-LDQ-	ADVP-PUT	PRPS
-LRB-	ADVP-PUT-TPC	RBR
-RDQ-	ADVP-TMP-TPC	RBS
-RRB-	ADVP-TPC	ROOT
ADJP-ADV	CC	S-VP
ADJP-HLN	CD	SYM
ADJP-LOC	CONJP	UCP-DIR
ADJP-MNR	EX	UCP-LOC-PRD
ADJP-PRD-TPC	FRAG-PRD	UCP-MNR
ADJP-SBJ	FRAG-SBJ-TTL	UCP-TPC
ADJP-TMP	FRAG-TTL	UH
ADJP-TPC	FW	VP-TPC
ADJP-TTL	INTJ	WDT
ADVP-CLR-DIR	INTJ-CLR	WHADJP
ADVP-CLR-LOC	INTJ-HLN	WHADVP
ADVP-CLR-LOC-TPC	JJR	WHADVP-TMP
ADVP-CLR-MNR	JJS	WP
ADVP-CLR-TMP	LS	WP$
ADVP-CLR-TPC	LST	WRB
ADVP-DIR	NAC	X-ADV
ADVP-DIR-TPC	NAC-LOC	X-HLN
ADVP-HLN	NAC-TMP	X-TTL
ADVP-LOC-PRD-TPC		

APPENDIX B

EMPTY CATEGORIES IN THE FEATURE-CONSTRAINT GRAMMAR AND PCFG

Table B.1 lists the empty categories in our grammar. Empty categories in our grammar are flanked by the + symbol. Translations from our notation to Penn Treebank null categories are noted earlier in Chapter 2, Table 2.6. In general, a category like +T-NP+ translates to the Penn Treebank category NP *T*. +C+ is the empty complementizer.

Table B.1: List of empty categories in the feature-constraint grammar.

category +C+ ;	category +RNR-ADJP+
category +E-NP+ ;	category +RNR-NP+
category +EI-NP+ ;	category +RNR-NX+
category +EI-PP+ ;	category +RNR-PP+
category +ELL-ADJP+ ;	category +RNR-S+
category +ELL-ADVP+ ;	category +RNR-SBAR+
category +ELL-NP+ ;	category +RNR-VP+
category +ELL-PP+ ;	category +T-ADJP+
category +ELL-S+ SLASH Slash; ;	category +T-ADVP+
category +ELL-SBAR+ ;	category +T-ADVP-VP+
category +ELL-VP+ ;	category +T-FRAG+
category +EXP-S+ ;	category +T-NP+
category +EXP-SBAR+ ;	category +T-NP-VP+
category +ICH-ADJP+ ;	category +T-PP+
category +ICH-ADVP+ ;	category +T-S+
Continued on next page	

Table B.1 – continued from previous page	
category +ICH-FRAG+ ;	category +T-S-SBAR+
category +ICH-NP+ ;	category +T-SBAR+
category +ICH-NX+ ;	category +T-SBARQ+
category +ICH-PP+ ;	category +T-SINV+
category +ICH-QP+ ;	category +T-SINV-SBAR+
category +ICH-S+ ;	category +T-SQ+
category +ICH-SBAR+ ;	category +T-UCP+
category +ICH-SBARQ+ ;	category +T-VP+
category +ICH-VP+ ;	category +U+
category +NIL+ ;	category +WHADVP+
category +NOT-ADVP+ ;	category +WHNP+
category +NOT-PP+ ;	category +WHPP+
category +NOT-SBAR+ ;	category +T-SBARQ-SBAR+
category +PPA-ADVP+ ;	category +T-SBAR-SBAR+
category +PPA-NP+ ;	category +T-FRAG-SBAR+
category +PPA-PP+ ;	category +ICH-UCP+
category +PPA-S+ ;	category +EXP-ADJP+
category +PPA-SBAR+ ;	category +ELL-X+
category +E-PP+ ;	

Table B.2: Variables used in the the feature-constraint grammar.

SLASH	sl
VFORM	st
VFORM	vf
PREP	pr
PERP	pp
PRTCL	pt
SBJ	sb
PERCENT	pc
UP	upp
NVAL	nv
PARENT	par

Table B.3: A few dummy rules added to the feature-constraint grammar in addition to Penn Treebank rules.

X	– > ‘X +EI-PP+
X	– > ‘X +ELL-ADVP+
X	– > ‘X +NOT-ADVP+
X	– > ‘X +NOT-PP+
X	– >‘X +NOT-SBAR+
X	– > ‘X +PPA-ADVP+
X	– > ‘X +PPA-S+
X	– > ‘X +RNR-ADJP+
X	– > ‘X +T-VP+
X	– > ‘X +EI-NP+

APPENDIX C

PARSEVAL RESULTS OF PCFGS

This chapter documents the PARSEVAL scores of PCFG with specific prepositions incorporated into verbal and nominal categories. Table C.1 shows the details of different PCFG versions and Table C.2 shows labeled bracketing results on Testset II. The differences between the models are not statistically significant, except that precision for model a19v5 is significantly worse than that for model a19v1 ($p < 0.005$). Table C.3 shows PARSEVAL scores of PCFGs trained with different amounts of training data.

Table C.1: Five different grammar versions with specific prepositions incorporated into verbal and/or nominal categories, with the number of non-lexical rules in each.

Version	Number of grammar rules (excluding pre-terminal)	Features
a19v1	66859	no prepositions in corporated in either verb or noun tags
a19v2	93715	prepositions on nouns (NN, NNS) but not verbs
a19v3	66859	prepositions on verb and but not nouns
a19v4	93506	prepositions both on verbs and nouns (NN, NNS)
a19v5	95449	prepositions on verbs and nouns (NN, NNS, NNP, NNPS)

Table C.2: Evalb results on heldout data (half of Testset II) for different grammar versions with or without prepositions incorporated on nominal and verbal categories. See Table C.1 for a list of the categories in which prepositions are incorporated for each grammar version.

	a19v1	a19v2	a19v3	a19v4	a19v5
– All –					
Number of sentence	2254	2254	2254	2254	2254
Number of Error sentence	7	7	7	7	7
Number of Skip sentence	0	0	0	0	0
Number of Valid sentence	2247	2247	2247	2247	2247
Bracketing Recall	86.52	86.7	86.51	86.72	86.62
Bracketing Precision	86.94	86.68	86.9	86.73	86.46
Bracketing FMeasure	86.73	86.69	86.71	86.72	86.54
Complete match	31.11	31.73	31.15	31.64	31.15
Average crossing	1.35	1.37	1.36	1.36	1.4
No crossing	59.28	59.19	59.23	59.37	58.88
2 or less crossing	81.4	81.09	81.53	81.35	81
Tagging accuracy	96.95	96.95	96.93	96.91	96.91
– len<40 –					
Number of sentence	2088	2088	2088	2088	2088
Number of Error sentence	7	7	7	7	7
Number of Skip sentence	0	0	0	0	0
Number of Valid sentence	2081	2081	2081	2081	2081
Bracketing Recall	87.47	87.57	87.46	87.66	87.58
Bracketing Precision	87.92	87.66	87.88	87.75	87.53
Bracketing FMeasure	87.69	87.61	87.67	87.71	87.55
Complete match	33.35	34.12	33.4	34.02	33.49
Average crossing	1.1	1.11	1.1	1.1	1.12
No crossing	62.81	62.85	62.71	63.05	62.66
2 or less crossing	84.86	84.38	84.96	84.72	84.43
Tagging accuracy	96.95	96.94	96.93	96.91	96.92

Table C.3: Evalb results on heldout data for grammars trained with different amounts of training data. Version t0 has the same incorporated feature set as version a19v1, but has every 10th sentence from the treebank held-out and all occurrences of about 112 mid-frequency verbs (about 1200 sentences) held out. The scores therefore are not identical to the ones for a19v1 in other tables.

Number of training sent.	38000	24308	16221	8097	4054
Version name	t0	t1	t2	t3	t4
– All –					
Number of sentence	3879	3878	3879	3875	3859
Number of Error sentence	13	14	13	12	13
Number of Skip sentence	0	0	0	0	0
Number of Valid sentence	3866	3864	3866	3863	3846
Bracketing Recall	85.65	85.18	84.62	83.14	81.24
Bracketing Precision	86.14	85.33	84.35	81.98	78.9
Bracketing FMeasure	85.89	85.26	84.49	82.56	80.06
Complete match	31.58	30.67	28.84	25.63	22
Average crossing	1.35	1.42	1.53	1.78	2.1
No crossing	60.14	59.03	57.19	52.47	49.3
2 or less crossing	81.89	80.23	78.92	75.59	72.31
Tagging accuracy	97.09	96.87	96.89	96.69	96.43
– len< 40					
Number of sentence	3582	3582	3582	3578	3563
Number of Error sentence	12	12	12	11	11
Number of Skip sentence	0	0	0	0	0
Number of Valid sentence	3570	3570	3570	3567	3552
Bracketing Recall	86.75	86.24	85.77	84.47	82.59
Bracketing Precision	87.26	86.44	85.61	83.42	80.33
Bracketing FMeasure	87.01	86.34	85.69	83.94	81.45
Complete match	34.01	33.03	31.06	27.73	23.82
Average crossing	1.07	1.14	1.21	1.42	1.71
No crossing	63.73	62.66	60.92	56.15	52.98
2 or less crossing	85.32	83.75	82.41	79.67	76.49
Tagging accuracy	97.06	96.86	96.89	96.71	96.45

APPENDIX D

SUBCATEGORIZATION FRAMES IN THE PCFG

This appendix lists all subcategorization frames that have occurred with verbs in Testset II, illustrated with a few example sentences from the Penn Treebank for each verb. The total number of verbs in Testset II is 11710 verb tokens and 2793 verb types. Penn Treebank annotation is not shown in most cases for readability, except for empty categories, unless the structure is rare. The column titled **Rule shape** denotes the immediate complement of the verb (sister of the verb). In the example sentences, the verb in question is shown in boldface. The subcategorization frame consists of the features `Val.Sbj.Vsel`. See Chapter 2 Table 2.1 for descriptions of different values of the Valence (`Val`) feature. Values of `Vsel` and `Sbj` along with descriptions are shown below for convenience.

Vsel		Sbj	
fin	tensed verb	e	Trace of A movement
base	base form		+E-NP+, +EI-NP+, etc.
to	TO	t	Trace of A-bar movement
n	passive past participle		+T-NP+, +T-S+, etc.
h	non-passive past participle	-	Non-empty subject
g	gerund		
sc	small clause		
scclr	closely related small clause		

Table D.1: Subcategorization frames in the PCFG

Frame	Freq.	Rule shape	Example PTB sentences.
n.-.-	3780	NP	**bid** one yen in two separate public auctions since 1987.
t.-.-	1134	-PRD	**are** at seasonally adjusted annual rates. **is** already under attack for excesses elsewhere. **is** the biggest problem facing the airline industry.
z.-.-	999	intransitive	**died** at 52 in an automobile accident.
b.-.-	990	SBAR	**complain** that they have limited access to government procurement in Japan.
aux.-.n	700	VP	**was** kidnapped +E-NP+ into this country. **was** pushed +E-NP+ up by tax-rate reduction. **was** not fully financed +E-NP+.
np.-.-	604	NP PP-CLR	**provide** an awareness to citizens and lawmakers.
aux.-.h	518	VP	**have** been invited to screenings. **has** also encouraged his staff.
p.-.-	516	PP-CLR	**apologizing** for his company 's making bids of just one yen for several government projects.
s.e.to	495	S	**want** to communicate to his partner. **seemed** to hint at that.
aux.-.g	376	VP	**was** offsetting purchases of mark and yen.
s.-.-	197	S	that got hard to take, he **added** +T-S+.
aux.-.base	145	VP	**does** not plan to be the leader. **do** think that acting out scripts is worth of CBS.
nr.-.-	138	PRT NP	**taking** on added risk. **warm** up the audience. **played** himself out.
d.-.-	136	PP-DIR	**migrate** to overseas exchanges such as London 's. **diversify** into dollar-denominated instruments.
de.-.-	109	NP-EXT PP-DIR	**advanced** 1 1/8 to 36 1/2. **skidded** 1.74 to 123.5 on turnover of 11 million shares.

Continued on next page

Table D.1 – continued from previous page			
Frame	**Freq.**	**Rule shape**	**Example PTB sentences.**
			fell 87.5 cents to $ 52.125 +U+.
dn.-.-	105	NP PP-DIR	**dumped** considerable energy into a whirling rampage through supermarket aisles. **accepting** substantial gifts from businessmen. **deliver** supplies to offshore rigs. **drive** some poorly managed properties into bankruptcy or new ownership.
s.-.to	102	S	**expect** them to blend the methodical marketing strategies.
s.-.sc	74	S	**make** funds available **pushing** labor ahead in the polls
m.-.-	68	NP NP	**ran** it the length of the South Gardens riverfront. **cost** the post office 10 million +U+ $16.1 +U+ in revenue in the past 12 months. **give** Greenspan a good rating.
e.-.-	53	NP-EXT, PRT NP-EXT	**turned** up a bit. **rose** 0.3 % in September to a $ 4.469 trillion +U+ rate. **edged** up 3.4 % to $ 904 million +U+ from $ 874 million.
s.e.sc	52	S	**declared** +E-NP+ effective. **looks** and **sounds** +E-NP+ forced. **considered** +E-NP+ a positive sign.
s.e.g	48	S	**begin** +E-NP+ mailing materials to shareholders at the end of this week. **continue** +E-NP+ pressing for early membership. **enjoy** +E-NP+ wallowing in such things.
r.-.-	48	PRT	**teaming** up **moving** in **calmed** down
bn.-.-	40	NP SBAR	**tell** them that they do n't have a job
s.-.base	37	S	**made** the landscape architects study a book on tartans. **hear** a Member of Congress moan about the deficit.
Continued on next page			

Table D.1 – continued from previous page			
Frame	**Freq.**	**Rule shape**	**Example PTB sentences.**
			helping her earn points in the state's incentive-bonus program.
s.-.fin	31	S	**acknowledged** : " it +EXP-SBAR+ 's never very clear who +T-NP+ starts what. **says** : " it 's big time chaos today. **adding**: "I want to help all".
ns.e.to	25	NP S	**persuade consumers** +E-NP+ to pay more than $ 15 +U+ for lipstick or eye makeup. **telling** her +E-NP+ to keep +E-NP+ sending things down to the lobby. **teach** managers +E-NP+ to accept reversals as a fact of business life.
aux.-.fin	21	VP	**has** not deemed any cases bad enough. he **has** said before that the country wants half the debt forgiven.
s.e.base	20	S	will **help**+E-NP+ meet increasing and diversifying demand. **help** +E-NP+ turn Southeast Asia into a more cohesive economic region.
pr.-.-	19	PRT PP-CLR	**clamping** down on campaigning. **ease** up on price-cutting. **fed** up with sackings of good people.
t.e.to	18	-PRD	**is** +E-NP+ to try +E-NP+ to get the law changed.
s.-.g	14	S	**set** the economy moving again. **keep** the government operating through Nov. 15. **featured** people holding up phone books. **left** observers wondering if it ever meant +E-NP+ to join.
npr.-.-	12	PRT NP PP-CLR	**give** up their baby for adoption. **farming** it out to law firms. **lock** in the difference in price as profit.
aux.-.-	7	VP	**is** (VP (IN about) (S +E-NP+ (VP (TO to) (VP (VB slip) (PP-DIR into recession)) were delivering goods more +ICH-SBAR+ in October than they **had** (VP +NIL+ (PP-TMP (for each of the five previous months)))
br.-.-	7	PRT SBAR	**find** out how your people are doing.
Continued on next page			

Table D.1 – continued from previous page			
Frame	**Freq.**	**Rule shape**	**Example PTB sentences.**
			figured out +C+ the Quotron numbers were wrong.
aux.-.wjj	7	PTB annotation error	(VP (VBD **were**) (VP (JJ crushed) +E-NP+)) (-PER- .)
br.-.-	7	PRT SBAR	**pointed** out that the recalls will have no impact on GE 's engine production.
s.t.sc	6	S	(NP (NNS stocks)) (SBAR +WHNP+ (S (NP-SBJ (PRP they)) (VP (VBP **keep**) (S +T-NP+ (PP-LOC-PRD (IN on) (NP (NN hand))
rs.e.to	5	PRT S	(VP (VBZ **has**) (PRT (RP yet)) (S +E-NP+ (VP (TO to) (VP (VB settle) (PP-CLR (IN with) (NP (DT either))) (PP (IN on) (NP (JJ new) (NNS contracts)
s.-.n	5	S	(VP (VBD did) (RB n't) (VP (NN **wish**) (S +E-NP+ (VP (TO to) (VP (VB be) (NP-PRD (DT a) (JJ full-time) (NN administrator)
aux.-.wnn	5	PTB annotation error	(VP (VBD **did**) (RB n't) (VP (NN wish) (S +E-NP+ (VP (TO to) (VP (VB be) (NP-PRD (DT a) (JJ full-time) (NN administrator)
en.-.-	5	NP NP-EXT	**lift** its production capacity 50 % **raise** its rates 5.3 % late this year or early next year **boosted** its quarterly dividend 18 %
ns.-.fin	4	NP S	**told** the Down Jones Industry report certainly, there are some who ...
bp.-.-	3	PP-CLR SBAR	**see** to it that their kids don't play truant **learned** of a coup plot that +T-NP+ might endanger his life
ep.-.-	3	NP-EXT PP-CLR	**shedding** 4.8 to 320
bp.-.-	3	PP-CLR SBAR	**confided** to investors that he had a secret agreement with Amoco Oil Co.
aux.-.to	3	VP	(VP (VBD **had**) (VP (TO to) (VP (VB defend) (NP (DT the) (NN company)) (PP-LOC (IN in) (NP (JJ such) (NNS proceedings))
rs.e.g	3	PRT S	**wind** up sharing the value of their concessions with public shareholders.

Continued on next page

Table D.1 – continued from previous page			
Frame	**Freq.**	**Rule shape**	**Example PTB sentences.**
ep.-.-	3	NP-EXT PP-CLR	**discounting** 75 % on an electronic evaluation of a well **rising** 2 1/4 to 43 1/4 on nearly eight million shares
s.e.wnn	2	PTB annotation error	(VP (VBZ **resembles**) (-LDQ- “) (S +E-NP+ (VP (NN juggling) (NP (NP (DT a) (JJ double-bladed) (NN ax)) (CC and) (NP (DT a) (NN buzz) (NN saw))
mp.-.-	2	NP PP-CLR NP, NP NP PP-CLR	**pay** Mr. Steinberg a premium for his shares.
ns.-.-	2	NP S	“You can see the highs and lows all here”, she **tells** the group +T-S+.
np.e.to	1	NP PP	**exceed** its assets by about 13.8 billion yen
rt.-.-	1	PRT -PRD	(VP (VB turn) (NP-PRD (DT the) (NN agency)) (PRT (RP around))
np.e.to	1	NP PP	**investigating** the bids for possible antitrust-law violations
bnp.-.-	1	NP PP-CLR SBAR	**take** it +EXP-SBAR+ as a sound bet that if it +EXP-S+ takes only 43 cents +E-NP+ to buy a dollar ’s worth of a firm ’s capital stock
t.-.fin	1	-PRD	**is** certain here: psyllium is likely to become the ...
s.-.wnn	1	S (annotation error)	(VP (VB **tolerate**) (S (NP-SBJ (DT the) (CD two) (NNP U.s.) (NN auto) (NNS giants))
s.e.wjj	1	S (annotation error)	(VP (VBZ **considers**) (S (NP-SBJ (DT the) (NN market)) (VP (JJ oversold) +E-NP+))
s.t.n	1	S	(NP-PRD (NP (NP (JJ numerous) (JJ other) (JJ impor tant) (NNS measures)) (SBAR +WHNP+ (S (NP-SBJ (NNPS Democrats)) (ADVP (RB badly)) (VP (VBD **wanted**) (S +T-NP+ (VP (VBN passed) +E-NP+))))))
ps.-.-	1	PP-CLR S	**shift** the burden +ICH-S+ to prosecutors +E-NP+ to disprove that discrimination caused a ny statistical racial disparities
ps.e.to	1	PP-CLR S	(VP (VBD **converged**) (PP-CLR (IN with) (NP (-LDQ- “) (NN mainstream) (-RDQ- ’”) (CC and) (NN demonizing))) (S +E-NP+ (VP (TO to) (VP (VB seal) (NP (NP (NNP Robert) (NNP Bork) (POS ’s)) (NN fate)))))
Continued on next page			

Table D.1 – continued from previous page			
Frame	**Freq.**	**Rule shape**	**Example PTB sentences.**
ns.e.base	1	NP S	(VP (VB **let**) (NP (PRP 's)) (S +E-NP+ (VP (VB rationalize) (NP (PRPS our) (NNS priorities))
ns.e.g	1	NP S	**spent** 40 years working to ensure that no such capitalist structures ever rose here.
ps.-.-	1	PP-CLR S	**called** upon +E-NP+ +E-NP+ to rescue the institution
rs.-.-	1	PRT S	(S (S-TPC (NP-SBJ (DT that)) (VP (VBZ cuts) (NP (DT the) (NN risk)))) (-COM- ,) (NP-SBJ (NP (NNP Mr.) (NNP Gregory)) (VP (VBZ **points**) (PRT (RP out)) +T-S+) (-PER- .)))

APPENDIX E

DISTRIBUTION

The programs used to build the augmented treebank and the treebank-aligned PCFG from the Penn treebank files, and some of the resulting resources, are being distributed in a release with the following functionality and/or components. The software environment and/or additional required software or databases are listed in parentheses.

1. Regularize the treebank. (lisp, awk, Penn Treebank II .mrg files)
2. Build feature constraint grammar from the output of 1. (perl, lisp)
3. Map each regularized treebank tree `ti.tb` to a trivial context free shared forest `ti.cpf` representing one tree. (lisp)
4. Solve feature constraints in each context-free shared forest `ti.cpf` to produce a feature shared forest `ti.fpf`. (yap-compiler, yap-parser, (Schmid, 2000b))
5. Map feature shared forests to PCFG rules and lexical entries with incorporated features. Parameter files for several choices of incorporations are included. (yappffun (Privman, 2003), java)
6. Adapt PCFG lexicon to a test corpus by tagging the test corpus and smoothing the PCFG lexicon to include the word forms in the test corpus (Treetagger, (Schmid, 1994), perl).
7. PCFG Viterbi parsing with labeled bracket and valence evaluation (bitpar (Schmid, 2004), evalb, perl)
8. Lexicon smoothing for modified inside-outside procedure and re-estimation on unsupervised training corpus (perl, bitpar).
9. Constraint grammar files that are the output of 3 for trees in Treebank II sections 0-15 whose index modulo 10 is not 9.

10. PCFG grammar and lexicon files with incorporated features for parsing sentences in PTB sections 0-22 with index 9 modulo 10.
11. Smoothed PCFG lexicon file with incorporated features for parsing 4 million word WSJ corpus.
12. Re-estimated PCFG lexicon with incorporated features for 4 million word WSJ corpus.

The experiment is organized with a make file which allows the experiment to be built from the treebank distribution. The modular components can be used in ways other than the ones discussed here. For instance, feature constraints can be solved in the maximal probability tree resulting from Viterbi parsing or each each of the N trees with highest PCFG probability. This allows feature trees to be constructed for novel data.

BIBLIOGRAPHY

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

Baker, J. K. (1979). Trainable grammars for speech recognition. In Klatt, D. & Wolf, J. (Eds.), *Speech Communication Papers for ASA'97*, pp. 547–550.

Bangalore, S. & Joshi, A. K. (1999). Supertagging: An Approach to Almost Parsing. *Computational Linguistics*, 25:237–265.

Basili, R., Pazienza, M. T., & Vindigni, M. (1997). Corpus-driven unsupervised learning of verb subcategorization frames. In Lenzirini, M. (Ed.), *Advances in Artificial Intelligence*, no. 1321 in Lecture Notes in Arti cial Intelligence. Springer-Verlag.

Beil, F., Carroll, G., Prescher, D., Riezler, S., & Rooth, M. (1999). Inside-outside estimation of a lexicalized PCFG for German. In *Proceedings of the 37th Annual Meeting of the Association of Computational Linguistics*.

Bikel, D. (2005). *On the Parameter Space of Generative Lexicalized Statistical Parsing Models.*. Ph.D. thesis, University of Pennsylvania.

Brent, M. (1991). Automatic acquisition of subcategorization frames from untagged text. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*.

Bresnan, J. (2001). *Lexical Functional Syntax*. Blackwell.

Briscoe, T. & Carroll, J. (1997). Automatic Extraction of Subcategorization from Corpora. In *Proceedings of the 5th ACL Conference on Applied NLP*.

Carroll, G. & Rooth, M. (1998). Valence induction with a head-lexicalized PCFG. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP) 1998*.

Carroll, J., Minnen, G., & Briscoe, E. (1998). Can subcategorization probabilities help parsing. In *6th ACL/SIGDAT Workshop on Very Large Corpora*. Montreal, Canada.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*. Menlo Park: AAAI Press/MIT Press.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*.

Chomsky, N. (1995). *The Minimalist Program*. Cambridge, MA.: MIT Press.

Clark, S. & Curran, J. R. (2004). The Importance of Supertagging for Wide-Coverage CCG Parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pp. 282–288. Geneva, Switzerland.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of 35th Annual Meeting of the Association for Computational Linguistics*.

de Marcken, C. (1995). On the unsupervised induction of Phrase Structure grammars. In *Proceedings of the 3rd Workshop on Very Large Corpora*.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood estimation from incomplete data via the EM algorithm. *J. Royal Statistical Society,* 39(B):1–38.

Deoskar, T. (2008). Re-estimation of Lexical Parameters for Treebank PCFGs. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*.

Deoskar, T. & Rooth, M. (2008). Induction of Treebank-Aligned Lexical Resources. In *Proceedings of the 6th Language Resources and Evaluation Conference (LREC)*.

Eisner, J. (2001). *Smoothing a Probabilistic Lexicon Via Syntactic Transformations*. Ph.D. thesis, University of Pennsylvania.

Gazdar, G., Klein, E., Pullum, G. K., & Sag, I. (1985). *Generalized Phrase Structure Grammar*. Oxford: Blackwell.

Hara, T., Miyao, Y., & Tsujii, J. (2007). Evaluating Impact of Re-training a Lexical Disambiguation Model on Domain Adaptation of an HPSG Parser. In *Proceedings of the 10th International Conference on Parsing Technologies*.

Hindle, D. & Rooth, M. (1993). Structural ambiguity and lexical relations. *Computational Linguistics*, 18(2):103–120.

Hockenmaier, J. & Steedman, M. (2002). Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*.

Johnson, M. (1998). PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4).

Joshi, A. & Schabes, Y. (1997). Tree Adjoining Grammars. In Salomma, A. & Rosenberg, G. (Eds.), *Handbook of Formal Languages and Automata*. Heidelberg: Springer Verlag.

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3).

Klein, D. & Manning, C. (2003). Accurate unlexicalized parsing. In *Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo, Japan.

Korhonen, A. (2002). *Subcategorization Acquisition*. Ph.D. thesis, Univ. of Cambridge.

Lari, K. & Young, S. J. (1990). The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56.

Lease, M. & Charniak, E. (2005). Parsing Biomedical Literature. In Dale, R., Wong, K.-F., Su, J., & Kwong, O. (Eds.), *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP'05)*, vol. 3651 of *Lecture Notes in Computer Science*, pp. 58 – 69. Jeju Island, Korea: Springer-Verlag.

Magerman, D. M. & Marcus, M. P. (1990). Parsing a natural language using mutual information statistics. In *Proceedings of the American Association for Artificial Intelligence*, pp. 984–989.

Manning, C. (1993). Automatic acquisition of a large subcategorization dictionary from corpora. In *Proceedings of 31st Annual Meeting of the Association for Computational Linguistics*.

Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Miyao, Y., Ninomiya, T., & Tsujii, J. (2004). Corpus-oriented grammar development for acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In *Proceedings of IJCNLP-04*.

O'Donovan, R., Burke, M., Cahill, A., van Genabith, J., & Way, A. (2005). Large-Scale Induction and Evaluation of Lexical Resources from the Penn-II and Penn-III Treebanks. *Computational Linguistics*, 31:329–365.

Pereira & Schabes (1992). Inside-Outside re-estimation from partially bracketed corpora. In *Proceedings of 30th Annual Meeting of the Association for Computational Linguistics*. Newark, Delaware.

Pollard, C. & Sag, I. (1994). *Head Driven Phrase Structure Grammar*. Chicago, IL: CLSI/Chicago University Press.

Prescher, D. (2005). Inducing Head-Driven PCFGs with Latent Heads: Refining a Treebank Grammar for Parsing. In *Proceedings of the 16th European Conference on Machine Learning (ECML 2005), LNCS series.*.

Privman, L. (2003). Yappffun: Java implemention of shared forest algorithms. Computational Linguistics Lab, Cornell University.

Riezler, S., King, T. H., Kaplan, R. M., Crouch, R., Maxwell, J. T. I., & Johnson, M. (2002). Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pp. 271–278. Philadelphia, Pennsylvania, USA.

Rooth, M., Riezler, S., & Prescher, D. (1999). Inducing a Semantically Annotated Lexicon via EM-Based Clustering. In *Proceedings of 37th Annual Meeting of the Association for Computational Linguistics*.

Schmid, H. (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of International Conference on New Methods in Language Processing*.

Schmid, H. (2000a). LoPar: Design and Implementation. Arbeitspapiere des Sonderforschungsbereiches 340 149, IMS Stuttgart.

Schmid, H. (2000b). *YAP - Parsing and Disambiguation With Feature-Based Grammars*. Ph.D. thesis, University of Stuttgart.

Schmid, H. (2000c). *Yap Manual*.

Schmid, H. (2004). Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit Vectors. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*.

Schmid, H. (2006). Trace Prediction and Recovery with Unlexicalised PCFGs and Slash Features. In *Proceedings of 21st COLING and 44th Annual Meeting of the ACL*, pp. 177–184. Sydney, Australia.

Steedman, M. (1996). *Surface Structure and Interpretation*. Linguistic Inquiry Monograph. Cambridge, MA.: MIT Press.

Steedman, M. (2000). *The Syntactic Process*. MIT Press/Bradford Books.

Stowell, T. (1981). *Origins of Phrase Structure*. Ph.D. thesis, MIT.

Tsuruoka, Y. M., Yoshimasa & Tsujii, J. (2004). Towards efficient probabilistic HPSG parsing: integrating semantic and syntactic preference to guide the parsing. In *Proceedings of IJCNLP-04 Workshop: Beyond shallow analyses - Formalisms and statistical modeling for deep analyses*. Hainan Island, China.